ulm university universität ulm

Fakultät für Ingenieurwissenschaften und Informatik
Institut für Datenbanken und Informationssysteme

Masterarbeit
im Studiengang Medieninformatik

Konzeption und Realisierung einer mobilen Anwendung zur Untersuchung der menschlichen Lokalisationsfähigkeit

vorgelegt von

Marcel Erath

März 2017

1. Gutachter	Prof. Dr. Manfred Reichert
2. Gutachter	Dr. Rüdiger Pryss
Betreuer	Marc Schickler
Matrikelnummer	886592
Arbeit vorgelegt am	16.03.2017

Kurzfassung

Eine der wichtigsten Fähigkeiten des Menschen ist die Erkennung der Richtung aus welcher ein Geräusch zu hören ist. Mit Hilfe des Richtungshörens können wir uns sicher in unserer Umgebung bewegen oder uns auf bestimmte Geräusche oder Stimmen konzentrieren. Allerdings gibt es Menschen, die Schwierigkeiten und Probleme bei der Erkennung der richtigen Richtung haben oder von dieser Schwäche eventuell noch gar nichts wissen.

Innerhalb dieser Ausarbeitung wird die Konzeption und Realisierung einer mobilen Anwendung zur Untersuchung der menschlichen Lokalisationsfähigkeit vorgestellt. Mit dieser kann die eigene Leistung bei der Lokalisation von Geräuschen untersucht und auch trainiert werden.

Um die richtige Funktionsweise der Applikation und die Genauigkeit und Schnelligkeit beim Auffinden der Richtung eines Geräusches zu überprüfen, wurde eine Studie mit 24 Teilnehmern durchgeführt. In drei Durchgängen mussten sie die Position von jeweils unterschiedlich vielen Geräuschen bestimmen. Die Auswertung der Daten ergab, dass die eigene Lokalisationsfähigkeit mit Hilfe der mobilen Anwendung problemlos untersucht werden kann, allerdings mit älteren Anwendungen eine höhere Genauigkeit erzielt wurde.

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe. Sinngemäße Übernahmen aus anderen Werken sind als solche kenntlich gemacht und mit genauer Quellenangabe (auch aus elektronischen Medien) versehen.

Ulm, den 16.03.2017 Marcel Erath

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ANOVA Analysis Of Variance

HRTF Head Related Transfer Function

Hz Hertz

ILD Interaural Level Difference

IPD Interaural Phase Difference

ITD Interaural Time Difference

ms Millisekunden

MVC Model View Controller

PNG Portable Network Graphics

SDK Software Development Kit

SMS Short Message Service

VR Virtual Reality

1 Einleitung

Die räumliche Lokalisation von Geräuschen ist eine wichtige Fähigkeit des Menschen. Mit Hilfe dieser können wir uns besser in unserer Umwelt orientieren, Gefahren erahnen und ihnen aus dem Weg gehen oder uns auf ein besonderes Geräusch innerhalb unserer Umgebung konzentrieren. Allerdings kann das Richtungshören durch diverse Erkrankungen des Gehörs negativ beeinflusst werden. Dadurch ist unsere Sicherheit im Alltag nicht mehr gewährleistet. Außerdem gibt es Menschen, die sich zum Beispiel innerhalb eines mit mehreren Personen gefüllten Raums nicht auf eine bestimmte Stimme konzentrieren können und sich durch andere Stimmen oder Geräusche leicht ablenken lassen. Diese Tatsache kann eine erfolgreiche Kommunikation deutlich erschweren. Für Menschen, die Schwierigkeiten beim Richtungshören haben, wäre eine Möglichkeit zum Training der eigenen Lokalisationsfähigkeit enorm wichtig. Mit Hilfe einer mobilen Anwendung könnten außerdem das Verhalten und die erzielten Ergebnisse von betroffenen Personen gesammelt und gespeichert, später ausgewertet und für weitere Forschungen verwendet werden [46][49][48][50].

1.1 Ziel der Arbeit

Ziel dieser Arbeit ist die Konzeption und Realisierung einer mobilen Anwendung zur Untersuchung der menschlichen Lokalisationsfähigkeit. Im Vergleich zu ähnlichen, bereits entwickelten, Anwendungen wird dabei ein von Google bereitgestelltes SDK zur Erstellung und Positionierung von 3D-Sound verwendet. Mit der Verwendung dieser Bibliotheken wird eine genauere und schnellere Lokalisation von Geräuschen erwartet. Damit diese Erwartung überprüft werden kann, wird eine Studie durchgeführt, in welcher die Teilnehmer in drei unterschiedlichen Durchgängen die Anwendung testen. Dabei werden das Verhalten und die erzielten Resultate gespeichert und anschließend ausgewertet. Die App soll später auf einem iPhone oder iPad installiert werden können und

zum einen zur Untersuchung der eigenen Lokalisationsfähigkeit, zum anderen aber auch für das Training dieser verwendet werden können. Dadurch wird Menschen, die Probleme mit dem Richtungshören haben, eine Möglichkeit geboten, dieses zu verbessern. Zusätzlich können die aufgezeichneten Daten für weitere Forschungen im Bereich von Gehör-Erkrankungen genutzt werden.

1.2 Aufbau der Arbeit

Um einen besseren Überblick über den Aufbau dieser Arbeit zu bekommen, ist die gesamte Struktur in Abbildung 1.1 dargestellt. In einem ersten Kapitel werden zunächst die Grundlagen des menschlichen Richtungshörens sowohl aus medizinischer als auch aus technologischer Sicht betrachtet und bereits existierende Anwendungen, die ebenfalls 3D-Sound beinhalten, vorgestellt. Anschließend werden alle Anforderungen an die zu konzipierende und realisierende mobile Applikation aufgelistet und innerhalb einer Tabelle zusammengefasst. Dabei wird zwischen funktionalen und nicht-funktionalen Anforderungen unterschieden.

Abbildung 1.1: Aufbau dieser Ausarbeitung

Das erste Ziel dieser Ausarbeitung, die Konzeption der Anwendung, wird in Kapitel 4 erarbeitet und genauer erläutert. Diese wird vor allem durch unterschiedliche Diagramme und Grafiken veranschaulicht. Der Hauptteil beschäftigt sich mit der Implementierung der iOS App und insbesondere der Verwendung des bereits erwähnten SDKs. Hier werden außerdem aufgetretene Probleme und deren Lösungen thematisiert. Anschließend findet ein Abgleich der entwickelten mobilen Anwendung mit den zuvor erarbeiteten Anforderungen statt. In Kapitel 7 wird schließlich die Evaluierung der Anwendung näher erläutert. Die Ergebnisse einer durchgeführten Studie werden dabei mit denen aus früheren Ausarbeitungen mit bereits existierenden Anwendungen verglichen. Hier stellt sich heraus, ob durch die Verwendung des von Google bereitgestellten SDKs eine

genauere und schnellere Lokalisation von unterschiedlichen Geräuschen erzielt werden kann oder nicht. Am Ende dieser Thesis werden zunächst die Konzeption und die Entwicklung der Anwendung und vor allem die gesammelten Erkenntnisse in einem kurzen Abschnitt nochmals zusammengefasst. Zusätzlich werden diverse Weiterentwicklungs- und Verbesserungsmöglichkeiten der mobilen Anwendung vorgeschlagen.

2

Grundlagen des Richtungshörens

Die Fähigkeit des Richtungshörens ist für den Menschen von großer Bedeutung. Durch diese Begabung wird zum Beispiel ein sicheres Verhalten im Straßenverkehr ermöglicht. Es kann unter anderem erkannt werden, aus welcher Richtung eine Gefahr droht und dieser somit rechtzeitig aus dem Weg gegangen werden. Ein weiteres Beispiel für die Wichtigkeit des Richtungshörens ist die Orientierung. Blinde können sich dadurch in ihrer Umwelt orientieren und sich so besser zurecht finden.
Da die Lokalisation von Sound in der vorliegenden Arbeit eine sehr wichtige Rolle spielt, wird in diesem Kapitel erklärt, wie sie aus medizinischer Sicht funktioniert, wie sie technologisch umgesetzt werden kann und was es bereits für Anwendungen dafür gibt.

2.1 Richtungshören aus medizinischer Sicht

Bei der Lokalisierung eines Geräusches werden mehrere unterschiedliche Werte und Faktoren, wie zum Beispiel die Zeit oder die Intensität, in Betracht gezogen. Bereits 1907 wurde von Sir Rayleigh vermutet, dass die Unterscheidung von rechts und links durch die höhere Intensität der Empfindung an dem Ohr, das näher an einem Geräusch liegt, erklärt werden kann [41]. Mit der Durchführung von einigen Versuchen konnte er seine Vermutung bestätigen. Allerdings stellte er fest, dass die Lokalisation nur bei Tönen mit einer hohen Frequenz durch die Intensitätstheorie begründet werden konnte. Nach weiteren Versuchen, bei denen Stimmgabeln mit niedrigeren Frequenzen von 128 Hz verwendet wurden, stellte sich heraus, dass es Fälle gibt, bei denen die Lokalisierung mit Hilfe der Intensität nicht funktioniert. Laut Rayleigh muss in diesen Fällen eine Beurteilung mit Hilfe des Phasenunterschieds gemacht werden. Durch weitere Versuche wurde schließlich festgestellt, dass die Bestimmung der Position eines hohen Tons mit einer Frequenz von >512 Hz auf dem Intensitätsunterschied (ILD) basiert. Bei niedrigen Tönen mit einer Frequenz von <128 Hz basiert die Bestimmung auf dem Phasenunterschied

(IPD). Außerdem wurde festgestellt, dass keine genaue Aussage über die Position eines Tons getroffen werden kann, wenn sich dieser direkt vor bzw. hinter einer Person befindet. Die Ergebnisse von Sir Rayleigh wurden später als `Duplex Theorie` bezeichnet und durch zahlreiche psychophysische und physiologische Studien untermauert [35]. Stevens und Newman führten eine Studie durch, in welcher Töne durch einen Lautsprecher, der sich in einer Entfernung von 12 Fuß zum Teilnehmer[1] befand, abgespielt wurden [51]. Der Teilnehmer musste die Richtung des Tons hören und anschließend benennen. Die Ergebnisse stimmten mit der von Rayleigh aufgestellten `Duplex Theorie` überein. Mills führte 1958 ein weiteres Experiment durch, in welchem er die Hörschärfe maß [36]. Dabei wurden zwei Lautsprecher, zwischen welchen sich der Teilnehmer entscheiden musste, in der horizontalen Ebene positioniert. Dadurch wurde herausgefunden, dass die Herausforderung den kleinsten hörbaren Winkel zu finden mit dem Problem der `Duplex Theorie` bei der Bestimmung der Position eines Tons direkt vor oder hinter einer Person übereinstimmt.
Laut Trimble ist der Phasenunterschied bei der Lokalisierung eines Tons ein spezieller Fall des Zeitunterschieds (ITD) [52]. Dadurch kann die Lokalisierung auf zwei Faktoren reduziert werden, zum einen die Intensitätsdifferenz und zum anderen die Zeitdifferenz.

Interaural Level Difference

Die Pegeldifferenz oder auch Intensitätsdifferenz gibt den Unterschied der an beiden Ohren ankommenden Intensität einer Schallwelle an. Der Schall erscheint an dem Ohr, das näher zur Schallquelle liegt, lauter als an dem Ohr, das weiter entfernt ist. Die Differenz gibt einen ersten Hinweis zur Lokalisierung von Tönen mit einer hohen Frequenz in der horizontalen Ebene und basiert auf folgenden drei Mechanismen [58].

- Rückstreuung des Schalls von den Schultern oder dem Oberkörper zu den Ohren
- Streuung des Tons durch den Kopf oder die Sphäre
- Richtfaktor, der durch die Ohrmuschel erzeugt wird

Interaural Time Difference

Unter dem Begriff des Zeitunterschieds versteht man die unterschiedlichen Zeiten, in welcher eine Schallwelle an einem Ohr ankommt. Kommt der Schall zum Beispiel von links, so kommt er zuerst am linken Ohr und dann am rechten Ohr einer Person an.

[1] Aufgrund der besseren Lesbarkeit werden innerhalb dieser Ausarbeitung immer die männlichen Formen verwendet. Gemeint werden aber beide Geschlechter.

Diese Differenz ist vor allem zur Lokalisierung von Tönen mit einer niedrigeren Frequenz wichtig.

Wie bereits weiter oben erwähnt wurde, gibt es bei diesen beiden Faktoren allerdings Schwierigkeiten beim Lokalisieren einer Soundquelle, die sich direkt vor oder hinter einer Person befindet. Wightman und Kistler stellten fest, dass neben dem Zeit- und Intensitätsunterschied auch die Ohrmuschel wichtige Informationen zur Lokalisierung einer Schallquelle liefert [57]. Ältere Forschungen haben darüber hinaus ergeben, dass diese Informationen, die die Ohrmuschel liefert, ausschlaggebend für die Lokalisierung einer Soundquelle sind [24][25][42]. Außerdem wird das Problem vom Auffinden einer Soundquelle direkt vor oder hinter einer Person ebenfalls mit Hilfe von der Ohrmuschel bereitgestellten Hinweisen gelöst [13][38].
Zusammengefasst kann also gesagt werden, dass bei der Lokalisation eines Geräusches drei Faktoren eine wichtige Rolle spielen und wichtige Hinweise liefern. Diese sind die Interaural Level Difference, die Interaural Time Difference und die Ohrmuschel.

2.2 Richtungshören aus technologischer Sicht

Damit Anwendungen mit räumlichem Sound entwickelt werden können, muss zunächst das Abspielen eines Sounds in einem virtuellen Raum, zum Beispiel über Kopfhörer, umgesetzt werden. Noisternig et al. stellten eine Methode zur Reproduktion von 3D-Sound über Kopfhörer vor, in welcher sie einen Ansatz mit Ambisonic, das eine hohe rechnerische Effizienz ermöglicht, wählten [37]. Ambisonic ist ein Verfahren, das die Voraussetzungen schafft, räumlichen Sound in all seinen Dimensionen zu reproduzieren oder zu simulieren. Dabei ist es nicht auf eine bestimmte Anzahl an Übertragungskanälen begrenzt. Je höher die Anzahl, desto höher ist die zu erreichende richtungsabhängige Auflösung [18]. Nach Noisternig et al. erfordert die Simulation eines virtuellen akustischen Raums die Filterung des Impulses mit Hilfe der sogenannten `Head Related Transfer Function` (HRTF).
Die HRTF erfasst zum einen die Frequenz und zum anderen die Zeit der gehörten Hinweise bei der Lokalisation einer Sound-Position und wurde vor allem von Wightman und Kistler erforscht [56]. Sie erfasst Veränderungen einer Schallwelle auf dem Weg von der Quelle zu den Ohren einer Person. Diese Veränderungen können unter anderem Beugungen und Reflektionen sein, die durch Teile unseres Körpers, wie zum Beispiel

unseren Kopf, unsere Ohrmuscheln oder unsere Schultern, verursacht werden [39]. Die Berechnung der HRTF ist sehr aufwendig und muss sowohl für das linke als auch für das rechte Ohr separat berechnet werden [47].
Da Menschen ihre Lokalisationsfähigkeit verbessern können, indem sie leichte Kopfbewegungen machen, zeigten Begault und Wenzel, dass die Verfolgung von Kopfbewegungen in Systemen zur Sound-Reproduktion sehr wichtig zur Verbesserung der Lokalisationfähigkeit ist [12]. Aufgrund dieser Tatsache verwendeten Noisternig et al. bei der technischen Umsetzung von räumlichem Sound Head-Tracking-Geräte, die sie an den Kopfhörern anbrachten.
Das innerhalb dieser Arbeit verwendete Google VR SDK, näher vorgestellt in Abschnitt 5.1.3, verwendet ebenfalls die Ambisonic-Technologie [23]. Es umgibt den Zuhörer mit einer großen Anzahl von virtuellen Lautsprechern um Schallwellen aus einer beliebigen Richtung zu simulieren. Damit virtuelle Lautsprecher verwendet werden können, wird ebenfalls die HRTF verwendet. Außerdem wird auch von Google ein Head-Tracking-Verfahren benutzt, um eine genauere Lokalisation von Geräuschen zu ermöglichen. Dabei werden Rotationsinformationen vom Endgerät genutzt, um den Sound innerhalb der virtuellen Lautsprecher zu rotieren.
Ein anderer Ansatz zur technischen Umsetzung der Lokalisation eines räumlichen Sounds stellten Keyrouz und Diepold im Jahr 2006 vor. Sie entwickelten eine Technik, bei der zwei Mikrofone im Ohrkanal eines Roboterkopfes platziert wurden [28]. Im Vergleich zu früheren Arbeiten wurden hier anstatt eines ganzen Feldes von virtuellen Lautsprechern nur zwei verwendet. Durch die Verwendung eines vorgestellten Lokalisationsalgorithmus konnten so bessere Ergebnisse erzielt und die Anforderungen reduziert werden.

2.3 Anwendungen

Es gibt unterschiedlichste Anwendungen, bei denen 3D-Sound zum Einsatz kommt und eine wichtige Rolle spielt. Sánches und Sáenz stellten die Anwendung "AudioChile" vor, die eine virtuelle Umgebung beinhaltet, durch welche mit Hilfe von 3D-Sound navigiert werden kann [43]. Die Anwendung wurde entwickelt, um Kindern mit Sehschwäche beim Lösen von alltäglichen Problemen zu helfen. Dadurch sollte ihnen die Möglichkeit gegeben werden, das Leben, die Kultur und die Besonderheiten der verschiedenen Regionen in Chile erkunden zu können. In Abbildung 2.1 ist ein blindes Kind zu sehen, das gerade über eine Tastatur mit einem frühen Prototyp der Anwendung interagiert. Die Durch-

führung dieses Benutzerfreundlichkeitstests ergab unter anderem eine Rückmeldung zu abgespielten Sounds, welche für die Weiterentwicklung der Anwendung sehr wichtig war. Zum Beispiel wurde so herausgefunden, dass für jede Aktion ein auditives Feedback notwendig ist oder das Abspielen von Geräuschen unter anderem vor dem Erreichen einer Wand gut funktioniert hat.

Abbildung 2.1: Ein Blinder interagiert mit "AudioChile" [43]

Sánches entwickelte zusammen mit Lumbreras eine weitere Applikation namens "AudioDoom" [34]. Diese ermöglicht das Testen von kognitiven Aufgaben mit sehbehinderten Kindern. Während der Nutzung dieser Anwendung hält das Kind einen drahtlosen Joystick in der Hand, der die Interaktion mit und die Bewegung durch die bereitgestellte Umgebung ermöglicht. Außerdem wird dieses Eingabegerät zur Erkennung einer Sound-Quelle verwendet und muss in die Richtung der wahrgenommenen Position eines abgespielten Geräusches gehalten werden. Durch eine Studie konnte mit Hilfe dieser Anwendung festgestellt werden, dass eine Umgebung mit 3D-Sound blinden Kindern räumliche Strukturen ins Gedächtnis bringen kann.
Eine mobile App, mit der sich erste Erfahrungen mit 3D-Sound sammeln lassen, ist die von *DevelopmentSquared* entwickelte iOS App "3D-Sound-Illusionen" [16]. Dabei handelt es sich um eine sehr einfache Anwendung, innerhalb welcher man unterschiedlichste Geräusche auswählen und mit Hilfe von Kopfhörern in einem virtuellen Raum erleben kann. Im folgenden Screenshot der App ist eine Auswahl von möglichen Geräuschen zu sehen.

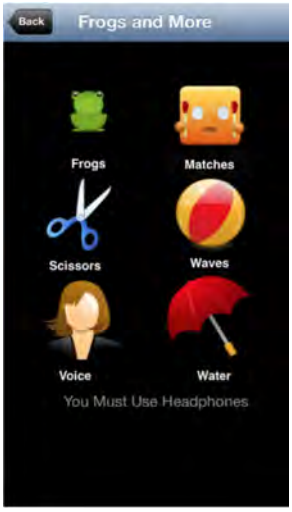
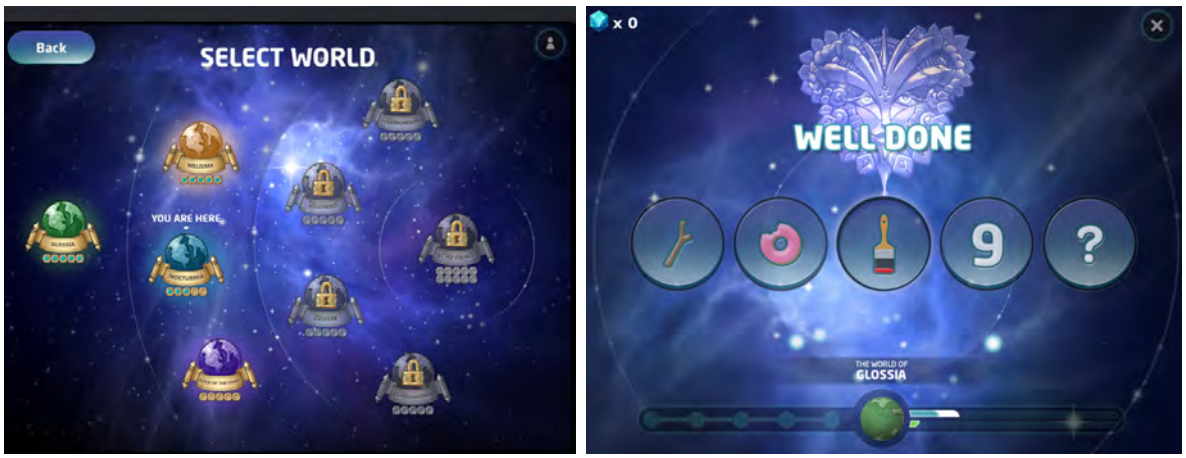


Abbildung 2.2: Screenshot der iOS App "3D-Sound-Illusionen" [16]

Eine weitere App namens "Sound Storm" wurde von *Australian Hearing Services* entwickelt und veröffentlicht [31]. Diese soll Kindern im Alter zwischen sechs und zwölf Jahren, die Hörschwächen in geräuschvollen Umgebungen wie zum Beispiel im Klassenzimmer haben, ein ungefähr dreimonatiges Training zur Unterdrückung von Hintergrundgeräuschen bieten. Dafür wird eine dreidimensionale auditive Umgebung bereitgestellt, durch welche sich die Kinder in zehn Level mit neun galaktischen Welten durcharbeiten müssen. Ein erster Eindruck der App kann durch folgende Abbildungen gewonnen werden. Hier ist zum einen die Levelauswahl und zum anderen der erfolgreiche Abschluss eines Levels zu sehen.



(a) Levelauswahl [32] (b) Erfolgreicher Abschluss eines Levels [33]

Abbildung 2.3: Screenshots der App "Sound Storm"

Breithut stellte in einem Artikel das mobile Spiel "Audio Defence: Zombie Arena" des Medienunternehmens *Somethin' Else* vor [14]. In diesem Spiel hört man über Kopfhörer Zombiegeräusche in einem virtuellen Raum. Um sich gegen die auftretenden Zombies zu wehren, muss das Smartphone in die jeweilige Richtung gehalten und eine Waffe abgefeuert werden. Aktuell (Stand Februar 2017) ist die iOS App im deutschen AppStore jedoch nicht mehr verfügbar. In Abbildung 2.4 sind Screenshots der Anwendung dargestellt. Im ersten Bild ist die Ansicht nach dem Abschluss einer Herausforderung zu sehen. Das zweite Bild stellt die Anzeige auf einem Endgerät während des Spielmodus dar. Um noch einen besseren Eindruck der Anwendung zu bekommen, kann auf der Videoplattform YouTube der offizielle Trailer zur Veröffentlichung der App angeschaut werden [44].

(a) Abschluss eines Levels [40] (b) Spielmodus [1]

Abbildung 2.4: Screenshots der App "Audio Defence: Zombie Arena"

Ein weiteres bereits im Jahr 2010 entwickeltes Spiel von *Somethin' Else* ist die iOS App "Papa Sangre". Kiss beschrieb diese als erste binaurale Echtzeit-3D-Audio-Engine, die auf einem tragbaren Gerät implementiert wurde [29]. Das Spiel wurde innerhalb von 73 Wochen entwickelt und bietet den Spielern fünf durchzuspielende Schlösser mit jeweils sieben Level und Missionen an. Das Ziel in jedem Level ist es, eine bestimmte Anzahl von Noten in Form von abgespielten Tönen zu sammeln und den anschließend erscheinenden Ausgang zu erreichen [54]. Während der Suche verfolgt ein Monster den Spieler, von welchem er sich nicht erwischen lassen darf. Die Fortbewegung innerhalb eines Levels erfolgt dabei mit abwechselndem Tippen auf zwei dargestellte Fußsymbole, welche in Abbildung 2.5 zu sehen sind. Ansonsten verzichtet die App weitestgehend auf grafische Elemente.

Die innerhalb dieser Arbeit vorgestellte Anwendung zur Untersuchung der menschlichen Lokalisationsfähigkeit ähnelt dem Prinzip der beiden zuletzt vorgestellten Spiele. Allerdings wird bei dieser Anwendung nicht komplett auf grafische Elemente verzichtet.

Abbildung 2.5: "Papa Sangre" im Spielmodus [55]

3

Anforderungen

In diesem Kapitel werden die Anforderungen an die zu entwickelnde mobile Anwendung aufgelistet und näher beschrieben. Dabei wird zunächst auf den geplanten Ablauf der Applikation eingegangen. Im nächsten Abschnitt werden unterschiedliche Konfigurationsmöglichkeiten der Anwendung erläutert. Bevor die vorgesehene Funktion zum Teilen von gespeicherten Daten erklärt wird, werden die Anforderungen an die Aufzeichnung von benötigten und wichtigen Daten beschrieben. Am Ende dieses Kapitels fasst eine Tabelle die zunächst definierten funktionalen und weitere nicht-funktionale Anforderungen zusammen.

3.1 Funktion der mobilen Anwendung

Der geplante Ablauf der Applikation kann am besten anhand eines Beispiels beschrieben werden. Angenommen man möchte heutzutage in der Natur spontan ein Tier fotografieren. Sofort wird das Smartphone hervorgeholt und die Kamera gestartet. Allerdings zeigt die Kamera kein Bild und auf dem Gerät ist nur ein schwarzer Bildschirm zu sehen. Um trotzdem ein Bild vom gesichteten Tier machen zu können ist der Verlass auf die Ohren unumgänglich. Durch Hören der Richtung aus welcher das Tier zu hören ist, kann das Smartphone in die jeweilige Position gebracht und ein Foto gemacht werden bevor das Tier abhaut. Dieses Beispiel spiegelt die wesentliche Funktion der zu entwickelnden App wider.
Bevor der Nutzer seine Lokalisationsfähigkeit jedoch testen kann, soll ihm zunächst in einem kleinen Menü die Möglichkeit zur Auswahl von unterschiedlichen Szenen geboten werden. Nachdem ein Szenario gewählt wurde, soll ein dunkler Bildschirm mit einem Auslösebutton für die Kamera angezeigt werden. Anschließend soll der Benutzer ein Tiergeräusch aus einer zufälligen Richtung hören, sich im besten Fall in diese Richtung drehen und die simulierte Kamera über den zur Verfügung gestellten Button auslösen.

Nach der simulierten Aufnahme eines Bildes soll im Hintergrund ein 360°-Bild der im Menü gewählten Szene und im Vordergrund, vorausgesetzt der Nutzer hat sich in die richtige Richtung gedreht, ein Foto des jeweiligen Tieres angezeigt werden. Falls das Smartphone in eine zur Position des Geräusches abweichende Richtung gedreht wurde, soll der Nutzer die Möglichkeit bekommen, sich in der Szene umzuschauen, um so die richtige Position des Tieres zu finden. Damit dieser Ablauf wiederholt werden kann, soll eine Option zum erneuten Starten der Untersuchung bereitgestellt werden.

3.2 Einstellungen der mobilen Anwendung

Damit unterschiedlichste Untersuchungen der menschlichen Lokalisationsfähigkeit getätigt werden können, sollen verschiedene Werte konfigurierbar sein. Um den Ablauf der Applikation zu verändern, sollen folgende Eigenschaften einstellbar sein:

- Lautstärke
- Genauigkeit
- Hintergrundgeräusch
- Erlaubte Zeit

Durch Veränderung des Lautstärkewertes soll eingestellt werden können, wie laut bzw. wie leise ein Tiergeräusch abgespielt werden soll. Der Wert Genauigkeit soll angeben, wie genau der Nutzer die Position des Tieres erkennen muss und soll als Winkel in Grad eingestellt werden. Mit der Option eines möglichen Hintergrundgeräusches soll untersucht werden können, ob ein Benutzer die Position des Tiergeräusches auch bei einem vorhandenen Hintergrundgeräusch erkennen kann. Die letzte Einstellung soll die Zeit in Sekunden angeben, die zum Finden der Position des Sounds höchstens benötigt werden darf.
Zusammengefasst soll die Konfiguration dieser Werte unterschiedlichste Schwierigkeitsstufen bei der Lokalisation eines Geräusches ermöglichen.

3.3 Datenspeicherung

Um später eine Auswertung des Nutzers ermöglichen zu können, sollen diverse Daten aufgezeichnet werden. Dafür sollen zum einen die Position, in die das Smartphone ge-

richtet ist, und zum anderen das Resultat der Lokalisierung eines Geräusches und die gewählten Einstellungen protokoliert werden. Folgende Daten sollen lokal in einer Text-Datei gespeichert werden:

- Position des Smartphones
  - x-Koordinate
  - y-Koordinate
  - z-Koordinate
- Ergebnis
  - gewählte Szene
  - Winkel zur Position des Tiergeräusches
  - benötigte Zeit
  - Art des Tiergeräusches
  - Position des Tiergeräusches
- Einstellungen
  - Lautstärke
  - Genauigkeit
  - Hintergrundgeräusch
  - Erlaubte Zeit

Dabei soll die Position des Smartphones ständig im Abstand von wenigen Millisekunden während der Untersuchung der Lokalisationsfähigkeit gespeichert werden. Das Ergebnis und die Einstellungen sollen beim Klick auf den Auslösebutton ausgelesen und in der Text-Datei vermerkt werden.

3.4 Teilen-Funktion

Abschließend soll die lokal gespeicherte Text-Datei ohne größeren Aufwand auf einen PC übertragen und dort ausgewertet werden können. Dafür soll die aus diversen anderen iOS Apps bekannte Teilen-Funktion verwendet werden. Diese Funktion ermöglicht das problemlose Versenden der aufgzeichneten Daten zum Beispiel per E-Mail.

3.5 Zusammenfassung

Alle soeben definierten funktionalen Anforderungen sind in Tabelle 3.1 zusammengefasst. Zusätzlich zu diesen werden auch diverse nicht-funktionale Anforderungen in dieser Tabelle veranschaulicht. Eine dieser nicht-funktionalen Anforderungen ist die einfache Steuerung der Anwendung insbesondere während der Untersuchung der Lokalisationsfähigkeit. Außerdem sollen alle Sounds schnell genug geladen werden, damit der Nutzer nicht zu lang auf das Abspielen eines Geräusches warten muss. Eine letzte nicht-funktionale Anforderung an die zu entwickelnde Anwendung ist die Verfügbarkeit der App sowohl für alle iPhone als auch für alle iPad Modelle.

Tabelle 3.1: Funktionale und nicht-funktionale Anforderungen an die zu entwickelnde App

Anforderung	**funktional**	**nicht-funktional**
App-Ablauf		
Menü zur Auswahl von unterschiedlichen Szenen	X	
Dunkler Bildschirm mit einem Auslöse-Button	X	
Abspielen eines Tiergeräusches aus einer zufälligen Richtung	X	
Darstellung eines 360°-Panorama-Bildes	X	
Darstellung des jeweligen Tieres	X	
Nutzer kann sich im Panorama-Bild umschauen	X	
Nutzer kann den Ablauf wiederholen	X	
Einfache Steuerung		X
Sounds sollen schnell genug geladen werden		X
App-Einstellungen		
Konfiguration des App-Ablaufs	X	
Lautstärke	X	
Genauigkeit	X	
Hintergrundgeräusch	X	
Erlaubte Zeit	X	
Daten-Logging		
Speichern der Position des Smartphones alle x Millisekunden	X	

Speichern des Ergebnises bei Klick auf den Kamera-Button	X	
Speichern der gewählten Einstellungen bei Klick auf den Kamera-Button	X	
Teilen-Funktion		
Möglichkeit zur Übertragung der gespeicherten Datei per E-Mail	X	
Sonstiges		
Verfügbarkeit für alle iPhone und iPad Modelle		X

4

Konzeption

Innerhalb dieses Kapitels werden im Abschnitt 4.1 einige Mockups dargestellt. Diese veranschaulichen einen ersten visuellen Entwurf der späteren iOS Applikation. Im nächsten Abschnitt wird zum einen das Datenmodell und zum anderen ein Komponentendiagramm gezeigt und genauer erklärt. Das Klassendiagramm der zu entwickelnden Anwendung wird im letzten Abschnitt dieses Kapitels dargestellt und erläutert.

4.1 Mockups

In diesem Abschnitt werden drei verschiedene Mockups vorgestellt. Mockups werden in einer frühen Phase der Entwicklung erstellt und veranschaulichen das spätere Look&Feel der Anwendung. Sie haben keine Funktionalität und dienen lediglich zur Visualisierung der Benutzeroberfläche. Alle Mockups wurden dabei mit Justinmind, einem freien Tool zum Entwickeln von Prototypen für alle möglichen Arten von Displays, entworfen [27].

4.1.1 Menü-Screen

Wie bereits in Abschnitt 3.1 beschrieben, soll der Nutzer der Applikation die Möglichkeit haben eine Szene wie zum Beispiel *Bauernhof* oder *Zoo* auswählen zu können. In Abbildung 4.1 ist zu sehen, dass dieses kleine Menü mit Hilfe einer Auflistung realisiert wird. Dabei wird die Auswahl auf der ganzen Displaygröße und jede einzelne Szene möglichst groß angezeigt, damit sich der Benutzer problemlos für ein Szenario entscheiden kann.

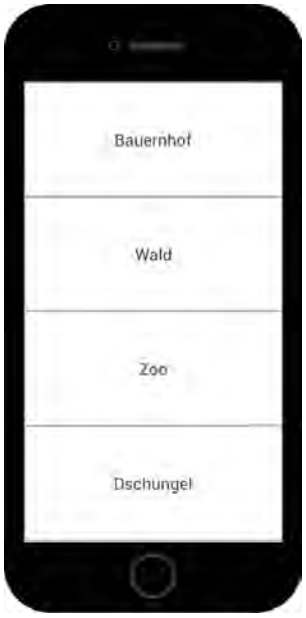


Abbildung 4.1: Menü-Screen zur Auswahl von verschiedenen Szenen

4.1.2 Foto-Screen

Der in Abbildung 4.2 dargestellte Foto-Screen wird dem Nutzer beim simulierten Fotografieren eines Tieres angezeigt. Da sich der Benutzer der App ausschließlich auf seine Ohren und nicht auf seine Augen verlassen soll, wird lediglich ein dunkler Hintergrund angezeigt. Damit schließlich ein Foto gemacht werden kann, wird ein zur iOS Kamera-App ähnlicher Button zur Verfügung gestellt. Wie in der Leitlinie *Händigkeit - Bedeutung und Untersuchung* zu lesen ist, wird vermutet, dass nur zwischen 10 und 15 % (Stand November 2014) der Bevölkerung Linkshänder sind [45]. Deshalb wird der Button zum Auslösen der Kamera am rechten Displayrand angeordnet.

Abbildung 4.2: Foto-Screen der iOS App

4.1.3 Einstellungen-Screen

In Abbildung 4.3 wird der Screen zur Konfiguration der eigentlichen Anwendung gezeigt. Wie zu sehen ist, hat der Nutzer hier die Möglichkeit, die Lautstärke des Tiergeräusches zwischen 1 und 5 einzustellen. Die Genauigkeit, mit der die Position des Geräusches erkannt werden muss, kann von *ungefähr* bis zu *sehr genau* variieren. Um die Schwierigkeit deutlich zu erhöhen, wird die Möglichkeit zum Einstellen eines Hintergrundgeräusches bereitgestellt. Außerdem wird ein, im Vergleich zu Kapitel 3.2 zusätzlicher, Parameter zur Konfiguration der Häufigkeit des abgespielten Tiergeräusches zur Verfügung gestellt. Durch diesen Wert kann die Schwierigkeit noch weiter verändert werden. Die letzte Einstellungsmöglichkeit ist die Angabe der erlaubten Zeit, die benötigt werden darf, um die Position des Geräusches zu lokalisieren. Am Ende des Screens ist ein Button zum Starten des in Abschnitt 3.1 beschriebenen App-Ablaufs angeordnet.

Abbildung 4.3: Einstellungen-Screen zum Konfigurieren des App-Ablaufs

4.2 Komponentendiagramm und Datenmodell

Dieser Abschnitt beinhaltet zum einen ein Komponentendiagramm, welches alle einzelnen Komponenten der gesamten Anwendung und deren Verbindungen untereinander darstellt. Zum anderen wird ein Datenmodell veranschaulicht, das vor allem die Struktur der zu entwickelnden iOS App beschreibt.

4.2.1 Komponentendiagramm

Wie in Abbildung 4.4 zu sehen ist, bildet die iOS App die zentrale Komponente der kompletten Anwendung. Um einen räumlichen Sound zu verwirklichen, wird das Google VR SDK für iOS für das Google Cardboard verwendet. Dieses SDK wird in Abschnitt 5.1.3 näher vorgestellt. Da auf der einen Seite die App Funktionen des SDK und auf der anderen Seite das SDK wiederum Funktionen von iOS verwendet, findet eine Kommunikation in beide Richtungen statt. Die Speicherung der Daten wird hier über eine Datenbank visualisiert. Daten werden zunächst von der App an einen Server gesendet. Dieser schickt wiederum eine Antwort an das Endgerät und speichert die Daten schließlich in einer Datenbank. Wie bereits in Kapitel 3 beschrieben wurde, soll die Datenspeicherung lediglich in Form einer Textdatei erfolgen. Da aber die Anwendung zu einem späteren Zeitpunkt um eine Datenbankanbindung zum Ablegen der Daten erweitert werden kann, wird in diesem Komponentendiagramm die Sicherung der Daten über eine Datenbank visualisiert.

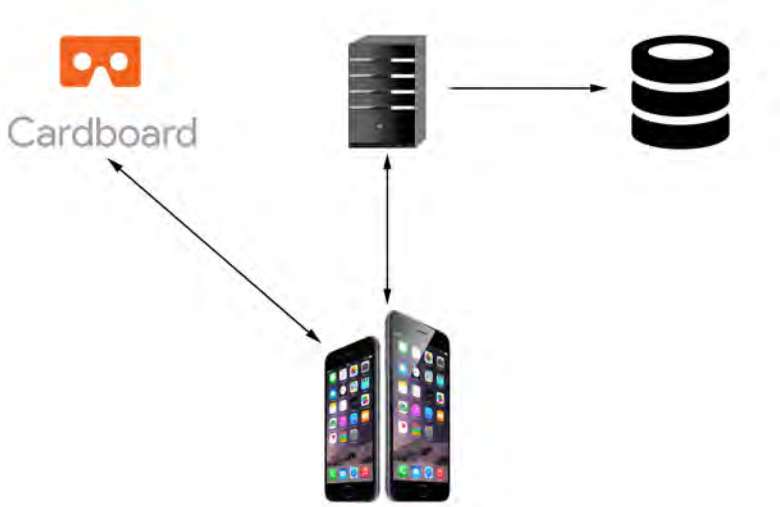


Abbildung 4.4: Komponentendiagramm der kompletten Anwendung

4.2.2 Datenmodell

In Abbildung 4.5 ist das Datenmodell der Anwendung zu sehen. Dabei sind wiederum die in Abschnitt 4.2.1 vorgestellten Komponenten zu finden, wobei das Hauptaugenmerk des Datenmodells auf dem Aufbau der iOS App liegt. Die App beinhaltet mehrere Szenen, die wiederum einige Tiere, ein 360°-Hintergrundbild und ein oder mehrere Hintergrund-

geräusch/e enthält. Jedes Tier besteht dagegen aus einem Bild, einem Geräusch und einer Position. Neben den Szenen enthält die Applikation zusätzlich die in Abschnitt 3.2 vorgestellten Einstellungen. Das Datenmodell des Servers und der Datenbank wird in dieser Arbeit nicht näher beschrieben, da diese zunächst nicht notwendig sind und nur für eine spätere Erweiterung der Anwendung gedacht sind. Auf das Google VR SDK wird, wie bereits erwähnt, in Abschnitt 5.1.3 näher eingegangen.

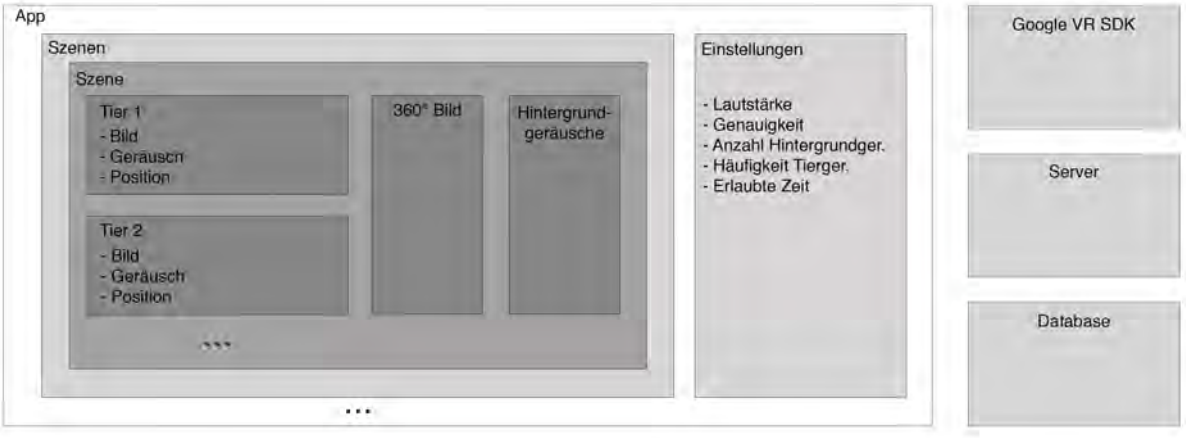


Abbildung 4.5: Datenmodell der gesamten Anwendung

4.3 Klassendiagramm

Die Anwendung wird mit Hilfe des MVC Architekturmusters entwickelt. Die Idee hinter diesem Muster besteht darin, die Daten (Modell) von der Benutzeroberfläche (View) zu trennen. Die Steuerung des Programms geschieht dabei durch den Controller [30]. Mit dieser Architektur können einzelne Komponenten in anderen Anwendungen wiederverwendet und die App problemlos erweitert werden. In Abbildung 4.6 ist das Klassendiagramm der zu entwickelten iOS Applikation zu sehen. Dabei sind aus Platzgründen nur die wesentlichsten Attribute und Methoden der einzelnen Klassen dargestellt. Das gesamte Klassendiagramm befindet sich im Anhang A. Allerdings wurde auch hier auf die Angabe aller Übergabeparameter der einzelnen Funktionen verzichtet.
Durch die drei dargestellten Screenshots wird die View-Ebene repräsentiert. Sie zeigen die Ansichten, für welche die darunter liegenden Controller verantwortlich sind. Die Hauptklasse der App bildet dabei die Klasse `SceneController`. Innerhalb dieser

befinden sich die meisten Funktionen der Anwendung. Hier werden alle 3D-Sounds erstellt und abgespielt, grafische Elemente erzeugt und dem Nutzer angezeigt sowie auf Eingaben und Aktionen des Benutzers reagiert. Zum Beispiel wird mit der Methode `createScene()` eine Szene und alle dazugehörigen Elemente erstellt. Die Szene wird nach der Lokalisation eines Geräusches mit der Funktion `updateScene()` aktualisiert. Durch den Aufruf von `showResult()` wird dem Benutzer der Anwendung das erzielte Ergebnis angezeigt. Die Klasse `SceneController` besitzt dabei ein oder mehrere Objekte der Klassen `SceneModel`, `AnimalModel`, `LogFileModel` und `SettingsModel`. Diese beinhalten alle notwendigen Daten, die für die Hauptfunktion der App notwendig sind.

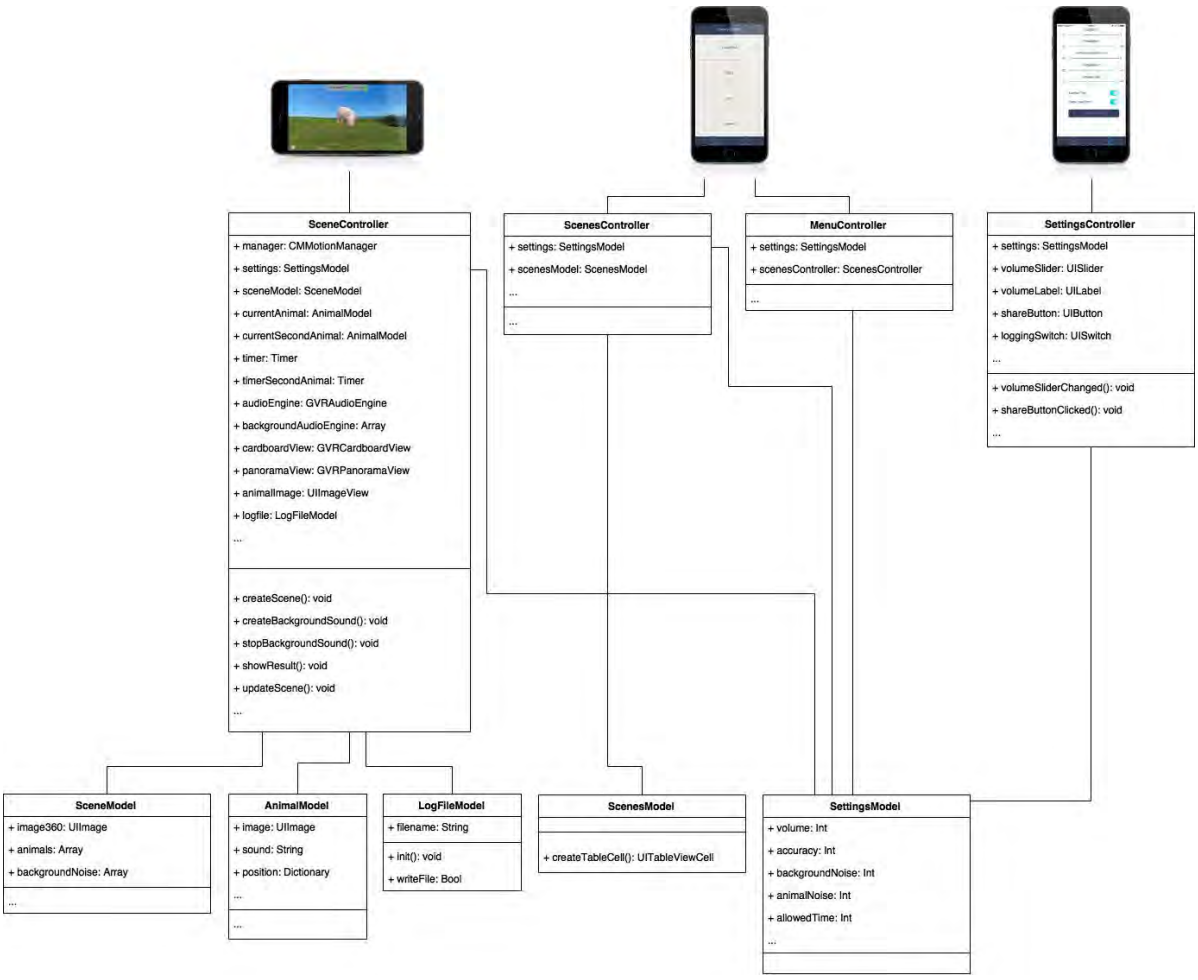


Abbildung 4.6: Klassendiagramm

Die weiteren Klassen `ScenesController` und `MenuController` sind für das Anzeigen der Auswahl der verschiedenen Szenen und der Menüleiste zuständig. Wie jede

Controller-Klasse besitzen auch diese ein `SettingsModel`-Objekt, welches die Daten der aktuellen Einstellungen enthält und beim Wechseln eines Views dem jeweiligen Controller übegeben wird. Mit der Methode `createTableCell()` der Klasse `ScenesModel` werden die einzelnen Zellen der Szenenauswahl erstellt.
Alle möglichen Änderungen der Einstellungen werden durch die Klasse `SettingsController` ausgeführt. Diese aktualisiert zum einen die Daten des `SettingsModel`-Objekts und zum anderen die grafischen Darstellungen. Für jeden konfigurierbaren Parameter gibt es je ein `UISlider`[2]- und ein `UILabel`[3]-Objekt und eine Methode, die für die Aktualisierung des Wertes zuständig ist.
Durch das dargestellte Klassendiagramm werden auch die drei unterschiedlichen Ebenen Model, View und Controller des Architekturmusters veranschaulicht. Hier ist gut zu sehen, dass nie ein View direkt mit einer Model-Klasse kommuniziert und die ganze Anwendung ohne große Herausforderungen erweitert werden kann.

[2]https://developer.apple.com/reference/uikit/uislider
[3]https://developer.apple.com/reference/uikit/uilabel

5 Implementierung

Dieses Kapitel beinhaltet die Beschreibung der Implementierung der iOS App zur Überprüfung der menschlichen Lokalisationsfähigkeit. Dabei werden in einem ersten Abschnitt die verwendeten Technologien vorgestellt. Darunter die Programmiersprache Swift von Apple und das Google VR SDK, welches zur Realisierung eines räumlichen Sounds verwendet wird. Anschließend geht der Hauptteil dieses Kapitels näher auf die Herausforderungen und aufgetretenen Probleme während der gesamten Programmierphase ein. Zum Abschluss wird in Abschnitt 5.3 der Testprozess der entwickelten Anwendung erläutert.

5.1 Technologien

Alle verwendeten Technologien werden in diesem Abschnitt vorgestellt. Der erste Teil beschäftigt sich mit der Programmiersprache Swift. Anschließend wird die benutzte Entwicklungsumgebung Xcode kurz beschrieben. Die letzte Passage dieses Abschnittes erklärt das von Google zur Verfügung gestellte SDK näher.

5.1.1 Swift

Apple veröffentlichte im Jahr 2014 die erste Version der Programmiersprache Swift. Diese ist eine leistungsstarke Sprache, die zum Entwickeln von Apps für iOS, Mac, Apple TV und Apple Watch verwendet werden kann [3]. Swift soll die früher verwendete Sprache Objective-C nicht ersetzen, sondern sich nahtlos in älteren Code einbinden lassen. Wichtige Merkmale sind unter anderem die Eigenschaften, dass Swift einfaches Lesen und Schreiben von Programmcode ermöglicht und dass der Speicher automatisch gemanagt wird. Apple zufolge besitzt die Programmiersprache zuätzlich folgende Eigenschaften [4]:

- Tupels und mehrfache Rückgabewerte
- Generische Typen
- Leistungsfähige Fehlerbehandlung
- Erweiterter Kontrollfluss mit den Schlüsselwörtern *do*, *guard*, *defer* und *repeat*
- Schnelle und präzise Iteration über eine Auswahl oder eine Kollektion
- u.v.a.

Um die Programmiersprache weiter zu verbreiten und jedem die Möglichkeit zur Entwicklung einer eigenen App zu bieten, veröffentlichte Apple mittlerweile die iPad App Swift Playgrounds [7]. Mit dieser App kann man Swift spielerisch und mit viel Spaß erlernen und dadurch auch den Einstieg in die Programmierung meistern.
Die aktuelle Version Swift 3.0 wurde im September 2016 veröffentlicht und wird zur Entwicklung der iOS Anwendung zur Untersuchung der menschlichen Lokalisationsfähigkeit verwendet [5].

5.1.2 Xcode

Xcode ist eine Entwicklungsumgebung von Apple, die kostenlos zum Download bereitgestellt wird. Diese stellt eine produktive Umgebung zur Entwicklung von diversen Apps für Mac, iPhone, iPad, Apple Watch und Apple TV zur Verfügung [8].

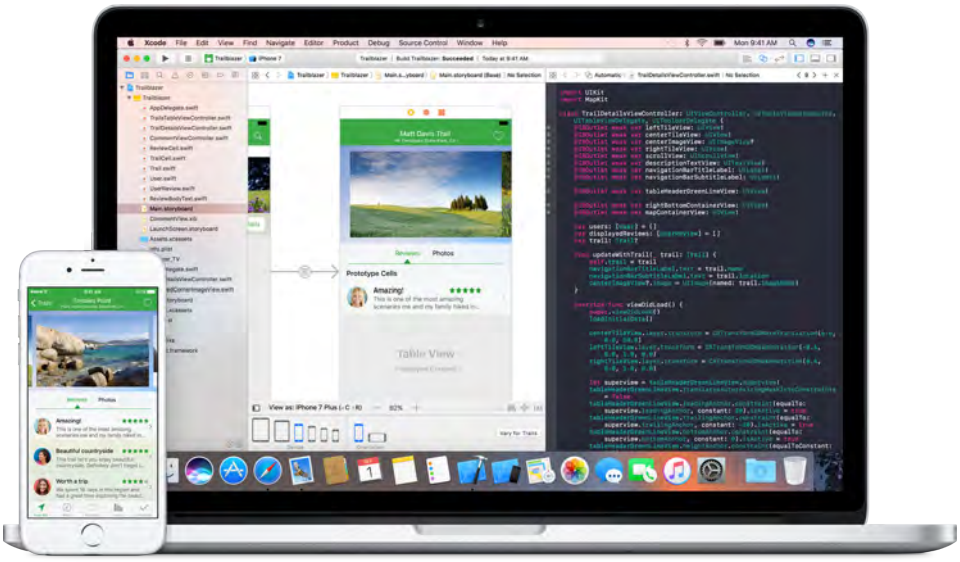


Abbildung 5.1: Beispielansicht von Xcode 8 [9]

In Abbildung 5.1 ist eine Beispielansicht von Xcode in der verwendeten Version 8 zu sehen. Dabei sind in der linken Spalte die Struktur und alle Dateien des aktuellen Projektes dargestellt. In der mittleren Spalte ist der sogenannte Interface Builder geöffnet. Dieser ermöglicht es, ohne Programmcode eine Nutzerschnittstelle zu erstellen. Dabei können grafische Elemente einfach per Drag&Drop zu einem Screen hinzugefügt und einzelne Screens miteinander verbunden werden. Mit Hilfe dieses Interface Builders ist es möglich, schnell und einfach Prototypen der späteren App zu erstellen. Die erzeugte Schnittstelle kann anschließend mit dem Quellcode, dargestellt in der rechten Spalte, verbunden und dadurch Funktionen zu den einzelnen Elementen hinzugefügt werden. Wie in der obigen Abbildungen zu sehen ist, lassen sich der Interface Builder und der Programmcode parallel öffnen. So wird eine übersichtliche und effiziente Programmierung ermöglicht.
Wird der Interface Builder allein geöffnet, werden einige wichtige Funktionen angezeigt. Wie in Abbildung 5.2 zu sehen ist, werden in der Objektliste (rot) zahlreiche Elemente aufgelistet, die auf einfache Weise auf einen Screen gezogen werden können.

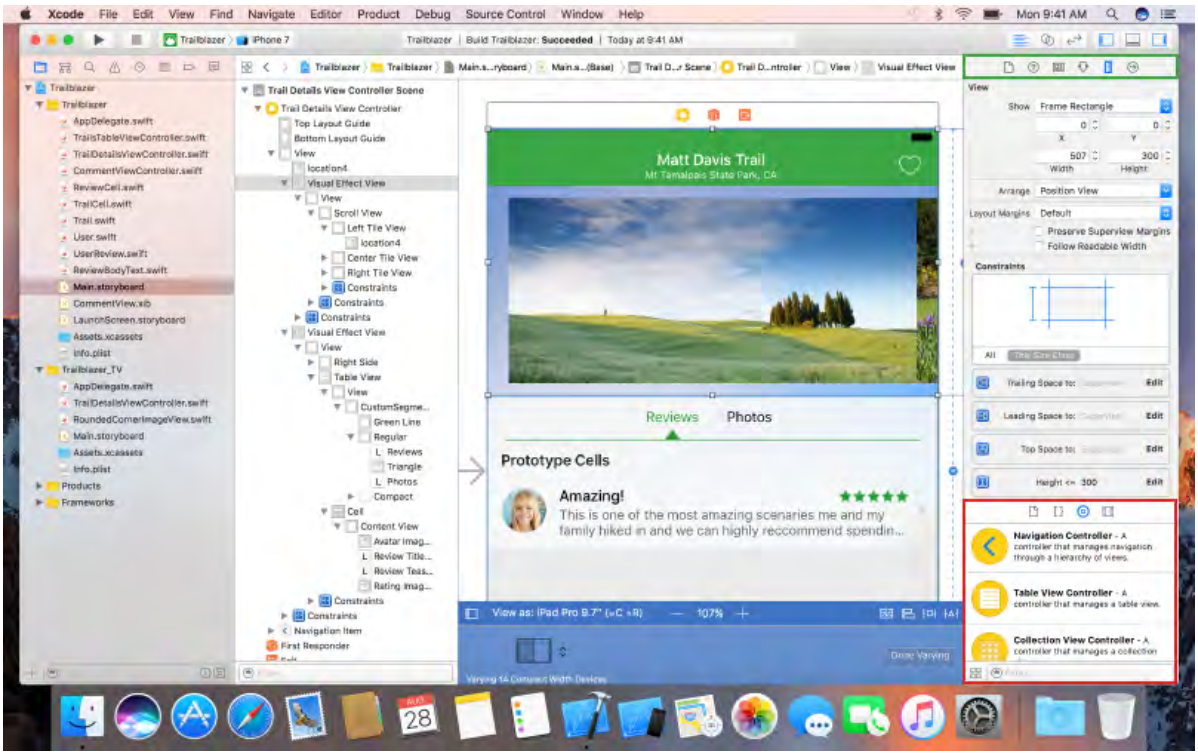


Abbildung 5.2: Interface Builder von Xcode 8 [10]

Außerdem sind in dieser Abbildung mehrere Werkzeuge dargestellt (grün), mit denen die einzelnen grafischen Elemente nach Belieben konfiguriert werden können. Diese Hilfsmittel sind von links nach rechts:

- File Inspector
- Quick Help Inspector
- Identity Inspector
- Attributes Inspector
- Size Inspector
- Connections Inspector

Darüber hinaus bietet Xcode noch zahlreiche weitere Funktionen wie zum Beispiel die Unterstützung des Versionsverwaltungssystems Git, die Verwendung von unterschiedlichen Schemas oder auch das schnelle Öffnen von gewünschten Dateien an.

5.1.3 Google VR SDK

Google bietet Entwicklern zwei Virtual Reality Plattformen an. Zum einen Google Cardboard und zum anderen Google Daydream [20][21]. Mit dem Cardboard stellt Google eine einfache und kostengünstige Lösung zum Erleben der virtuellen Realität zur Verfügung. Der Nutzer kann es selbst zusammenbauen und mit Hilfe seines Smartphones und zahlreicher Apps in virtuelle Welten eintauchen. Mit Daydream bietet Google eine Virtual Reality Lösung an, die bequemer zu tragen ist, eine höhere Qualität besitzt und mit einem zusätzlichen Controller verbunden werden kann. Allerdings kann diese Lösung bis jetzt nur mit wenigen auserwählten Smartphones, wie zum Beispiel den Google Pixel Modellen, verwendet werden.
Damit Entwickler zahlreiche Anwendungen entwickeln können, stellt Google für beide Plattformen SDKs für Unity, Unreal Engine, Android und iOS bereit. Wobei das iOS SDK nur Google Cardboard unterstützt. Mit Hilfe des iOS SDKs werden einige Herausforderungen für Virtual Reality Anwendungen vereinfacht [22]:

- Korrektur der Linsenverkrümmung
- räumliches Audio
- Verfolgung des Kopfes
- 3D-Kalibrierung

- Behandlung von Nutzereingaben
- u.v.a.

Wie bereits in Kapitel 3 beschrieben, ist das Abspielen von räumlichem Sound eine wichtige und dringend erfoderliche Anforderung an die zu entwickelnde App. Da das Google VR SDK unter anderem den Entwickler bei der Umsetzung von räumlichem Audio unterstüzt, wird dieses bei der Implementierung verwendet [23].

5.2 Programmierung

Dieser Abschnitt beinhaltet die Programmierung der iOS Anwendung und bildet einen wichtigen Teil dieser Ausarbeitung. Dabei wird die Entwicklung jedes einzelnen Screens und aller Funktionen näher beschrieben. Insbesondere auf die Realisierung des räumlichen Sounds wird genauer eingegangen.

5.2.1 Installation des Google VR SDKs

Damit das Google VR SDK installiert werden konnte, musste zunächst, wie von Google beschrieben, CocoaPods installiert werden [19]. CocoaPods ist ein Abhängigkeitsmanager für Swift und Objective-C Projekte und beinhaltet über 27.000 Bibliotheken [15]. Nachdem dieser Manager installiert wurde, konnte dem zuvor erstellten Xcode Projekt ein sogenanntes Podfile hinzugefügt werden. Da das Podfile alle Abhängigkeiten eines Projektes beinhaltet, musste das Google VR SDK in diese Datei eingefügt werden. Abschließend konnte das SDK mit Hilfe der Ausführung des folgenden Befehls innerhalb des Projektordners installiert werden.

Listing 5.1: Kommando zur Installation des Google VR SDKs

```
1  pod update
```

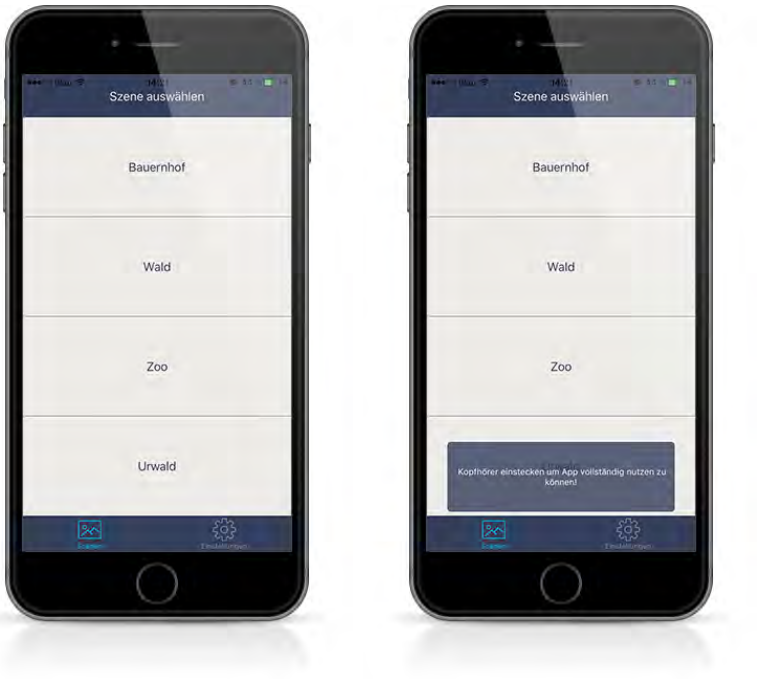
Da das Google SDK in Objective-C geschrieben ist und die iOS Anwendung in Swift programmiert wurde, musste ein Bridging Header zum Projekt hinzugefügt werden. Durch diese Datei werden Objective-C-Dateien zu einem Swift-Projekt hinzugefügt [6]. Dadurch wird die Verwendung von Objective-C-Funkionen innerhalb von Swift ermöglicht.

5.2.2 Menü

Wie in Abschnitt 4.1.1 bereits dargestellt wurde, sollte das Menü mit einer Auflistung realisiert werden. Um dies zu erreichen, wurde die Klasse `UITableViewController`[4] von der Klasse `ScenesController` vererbt. `UITableViewController` erzeugt ein Controller Objekt das eine Tabellenansicht verwaltet. Damit jede Tabellenzelle möglichst groß angezeigt werden kann, wird jedes Mal beim Öffnen des Menüs zunächst der zur Verfügung stehende Platz berechnet. Anschließend wird die entsprechende Höhe für jede Zelle kalkuliert. Dadurch wird sichergestellt, dass bei unterschiedlichen Displaygrößen jede Tabellenreihe möglichst groß dargestellt wird und so für den Nutzer leicht zu bedienen ist. Jedem einzelnen Tabellenelement wird dann eine Beschriftung, grafische Anpassungen und die exakte Höhe hinzugefügt. Nachdem ein Element erstellt wurde, wird es an die Tabellenansicht angehängt. Um dem Nutzer ein Feedback beim Auswählen einer Szene zu geben, werden jeder Tabellenreihe zusätzliche grafische Änderungen beigefügt. Damit der Anwender nach der Wahl der gewünschten Szene zum nächsten Screen gelangt, wird zunächst ein `SceneController`-Objekt erstellt. Diesem werden alle Szenen, der Index der gewählten Szene sowie alle Einstellungen übertragen. Durch die Übertragung dieser Daten kann anschließend die richtige Szene mit den zuvor konfigurierten Einstellungen abgespielt werden.
Um dem Nutzer eine möglichst einfache Bedienung der Anwendung zu ermöglichen, wurde zusätzlich eine Tabbar implementiert. Dabei erbt die Klasse `MenuController` von der Klasse `UITabBarController`[5]. Diese Klasse stellt am unteren Rand des Displays eine Menüleiste (Tabbar) zur Verfügung. Mit Hilfe dieser Leiste kann zwischen verschiedenen Screens gewechselt werden. Über den in Abschnitt 5.1.2 vorgestellten Interface Builder können unterschiedliche Screens zur Menüleiste hinzugefügt werden. Hier wurden zum einen die zuvor beschriebene Szenenauswahl sowie der, im nächsten Abschnitt erklärte, Einstellungen-Screen mit der Tabbar verbunden. Innerhalb der Klasse `MenuController` wird außerdem überprüft, ob der Nutzer Kopfhörer eingesteckt hat oder nicht. Ist dies nicht der Fall, wird er durch einen kleine angezeigte Meldung daran erinnert, dass die App nur mit der Verwendung von Kopfhörern vollständig genutzt werden kann. Außerdem wird nach dem Laden der Menüleiste, falls noch nicht vorhanden, ein Objekt der Klasse `SettingsModel` erzeugt. Durch Erstellung dieses Objektes wird sichergestellt, dass Einstellungen vorhanden sind und diese an andere Screens weitergegeben werden

[4] https://developer.apple.com/reference/uikit/uitableviewcontroller
[5] https://developer.apple.com/reference/uikit/uitabbarcontroller

können. In Abbildung 5.3 ist das gesamte Menü inklusive Menüleiste und Szenenauswahl zu sehen.



(a) Szenenauswahl

(b) Szenenauswahl mit angezeigter Meldung

Abbildung 5.3: Menüansicht der Anwendung

5.2.3 Einstellungen

Um dem Nutzer die Möglichkeit zur Konfiguration der App zu bieten sollte ein Screen mit diversen Einstellungen erzeugt werden. Dafür wurde die Klasse `SettingsController` erstellt, welche von der Klasse `UIViewController`[6] erbt. Die Klasse `UIViewController` wird zu Verwaltung eines Views verwendet. Damit dem Nutzer auch an dieser Stelle eine einfache Bedienung der Anwendung geboten werden kann, wurde die Einstellung der in Abschnitt 3.2 geforderten Parameter mit Hilfe eines Schiebereglers und nicht mit Hilfe von vielen kleineren Buttons umgesetzt. Hierbei wurde dem Einstellungen-Screen für jeden Parameter ein UISlider[7] mit Hilfe des Interface-Builders hinzugefügt und anschließend mit der Klasse `SettingsController` verbunden. Diese Klasse ist zum einen

[6]https://developer.apple.com/reference/uikit/uiviewcontroller
[7]https://developer.apple.com/reference/uikit/uislider

dafür zuständig, bei Änderung eines Reglers den Wert des jeweiligen Attributes des übergebenen `SettingsModel`-Objekts zu aktualisieren. Zum anderen dafür, dem Nutzer ein visuelles Feedback und einen aktualisierten Wert anzuzeigen. In der folgenden Abbildung ist der Einstellungen-Screen mit allen konfigurierbaren Parametern zu sehen.

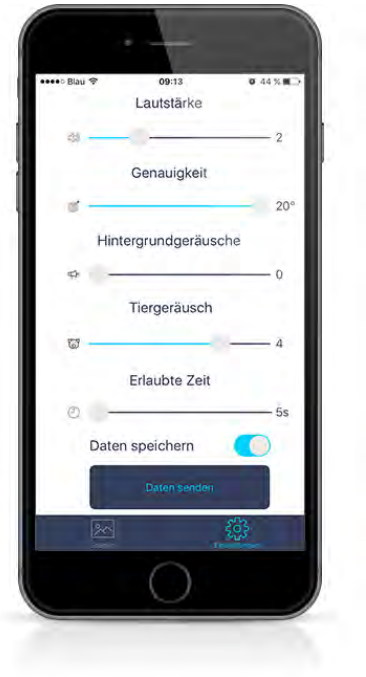


Abbildung 5.4: Entwickelter Einstellungen-Screen der iOS App

Wie in Abbildung 5.4 außerdem zu sehen ist, wird die geforderte Teilen-Funktion ebenfalls in diesem Screen angezeigt. Die Umsetzung dieser Funktion wird später in Abschnitt 5.2.6 näher beschrieben.
Vergleicht man den entwickelten Screen mit dem in Kapitel 4 dargestellten Mockup, fällt auf, dass der Start Button am unteren Rand des Screens fehlt. Da das Menü mit Hilfe einer Menüleiste und die Szenenauswahl mit einer Tabellenstruktur umgesetzt wurde, war dieser Button überflüssig. Man kann nun beliebig zwischen der Szenenauswahl und den Einstellungen wechseln. Beim Wählen einer Szene wird der geforderte App-Ablauf automatisch gestartet.

5.2.4 Ablauf der mobilen Anwendung

Dieser Abschnitt beinhaltet die Implementierung des geforderten Ablaufs der Anwendung. Da die Entwicklung dieses Ablaufs technisch am interessantesten ist und deshalb

genauer erklärt wird, wird dieser Abschnitt nochmals in mehrere Unterabschnitte unterteilt.

5.2.4.1 Implementierung eines Tieres

Wie im Klassendiagramm unter 4.3 zu sehen ist, wurde ein Tier mit Hilfe der Klasse `AnimalModel` implementiert. Jedes Tierobjekt besitzt dabei ein Bild, einen Tierlaut, eine Position und einen Namen. Um einen korrekten Ablauf der Anwendung zu garantieren, muss das Tierbild einen transparenten Hintergrund besitzen und im Bildformat PNG gespeichert sein. Außerdem ist zu beachten, dass das Tiergeräusch nur mit einer Tonspur aufgenommen wurde und somit mono ist.
Damit ein Objekt dieser Klasse erzeugt werden kann und alle Attribute richtig gesetzt werden können, wurde ein Konstruktor geschrieben, der den Tiernamen als Parameter übergeben bekommt. Bei der Initialisierung eines Tieres werden, je nach übertragenem Namen, alle Attribute mit den jeweiligen Werten belegt.

5.2.4.2 Implementierung einer Szene

Die Erstellung einer Szene wurde mit der Klasse `SceneModel` realisiert. Eine Szene hat als Attribute ein Bild, ein oder mehrere Tier/e und ein oder mehrere Hintergrundgeräusch/e. Um ein Objekt dieser Klasse zu erzeugen, wurde ebenfalls ein Konstruktor geschrieben, der als Parameter eine eindeutige Nummer zur Bezeichnung der jeweiligen Szene übergeben bekommt. Durch die jeweilige übertragene Nummer bei der Erstellung einer Szene werden die Attribute mit den richtigen Werten belegt beziehungsweise die zugehörigen Tiere erzeugt und zur Szene hinzugefügt.

5.2.4.3 Implementierung des Ablaufs

Nachdem im Menü eine Szene ausgewählt wurde, wird diese mit Hilfe der Klasse `SceneController` erstellt. Diese Klasse wurde so implementiert, dass zunächst die jeweilige Szene, wie im vorherigen Abschnitt beschrieben, erzeugt wird. Anschließend werden alle grafischen Elemente erstellt. Darunter eine Art Warnung, die den Nutzer auffordert sein Smartphone in die sogenannte *Landscape-Left*-Orientierung zu bringen, der zum Auslösen der simulierten Kamera benötigte Button sowie ein kleiner Button zum Zurückkehren in das Menü. Hält der Benutzer sein iPhone in der

Portrait-Orientierung, wird ihm die erzeugte und in Abbildung 5.5 dargestellte Meldung angezeigt. Befindet sich das Gerät bereits in der *Landscape-Left*-Orientierung oder wird in diese gebracht, startet die Untersuchung der Lokalisationsfähigkeit und dem Nutzer wird ein dunkler Bildschirm mit dem erstellten Kamerabutton angezeigt.

Abbildung 5.5: Aufforderung, das iPhone in die richtige Orientierung zu bringen

Um die gewünschte Funktion zu erreichen, war die Verwendung eines `GVRPanoramaView`[8] und eines `GVRCardboardView`[9] des Google VR SDKs notwendig. Aufgetretene Probleme und die daraus folgende Lösungen werden in den Abschnitten 5.2.4.4 und 5.2.4.5 näher beschrieben. Beide Views werden beim Start der Untersuchung zunächst konfiguriert. Anschließend wird mit Hilfe der von Google bereitgestellten Klasse `GVRAudioEngine`[10] eine Audio Engine zum Abspielen von räumlichem Sound erstellt. Da die Nutzung dieser Engine ein zentraler Punkt in dieser App ist, ist in Listing 5.2 die Implementierung der Erstellung und des Abspielens eines Tierlauts veranschaulicht.

[8]https://developers.google.com/vr/ios/reference/interface_g_v_r_panorama_view
[9]https://developers.google.com/vr/ios/reference/interface_g_v_r_cardboard_view
[10]https://developers.google.com/vr/ios/reference/interface_g_v_r_audio_engine

Listing 5.2: Erstellung eines `GVRAudioEngine` Objektes und Abspielen eines Sounds

```
// Create spatial sound
var audioEngine:GVRAudioEngine? = GVRAudioEngine(
  renderingMode: kRenderingModeBinauralHighQuality
)
let filePreloaded=audioEngine!.preloadSoundFile(animalSound)
audioEngine!.start()
soundID = -1
if (filePreloaded) {
  soundID = (audioEngine!.createSoundObject(animalSound))
  if (soundID != -1) {

    // Set sound position
    audioEngine!.setSoundObjectPosition(
      soundID,
      x: self.currentAnimal.position["x"]!,
      y: self.currentAnimal.position["y"]!,
      z: self.currentAnimal.position["z"]!
    )

    .....

    // Play sound
    self.playCounter = 0
    if ( self.settings.animalNoise > 1) {
      self.timer = Timer.scheduledTimer(
        timeInterval: 4,
        target: self,
        selector: #selector(self.playSound),
        userInfo: nil,
        repeats: true
      )
    } else {
      self.audioEngine!.playSound(
        self.soundID,
```

```
35          loopingEnabled: false
36        )
37      }
38    }
39  }
```

Dabei wird zunächst ein Objekt der Klasse `GVRAudioEngine` erstellt. Diesem wird der Renderingmodus `kRenderingModeBinauralHighQuality` übergeben. Durch diesen Modus wird ein Soundobjekt mit Hilfe von 16 virtuellen Lautsprechern simuliert [23]. Anschließend wird der jeweilige Tierlaut geladen und daraus ein Soundobjekt erzeugt. Diesem Soundobjekt werden dann die Koordinaten der Position des Tieres übergeben. Dadurch wird die Position des Sounds im 3D-Raum gesetzt. Da die Lokalisierung nur in der horizontalen Ebene erfolgen soll, wird die y-Koordinate des Soundobjektes immer auf 0 gesetzt. Für die x- und z-Koordinate werden Werte zwischen -1 und 1 verwendet. Das verwendete Koordinatensystem in der *Landscape-Left*-Orientierung ist in folgender Abbildung dargestellt.

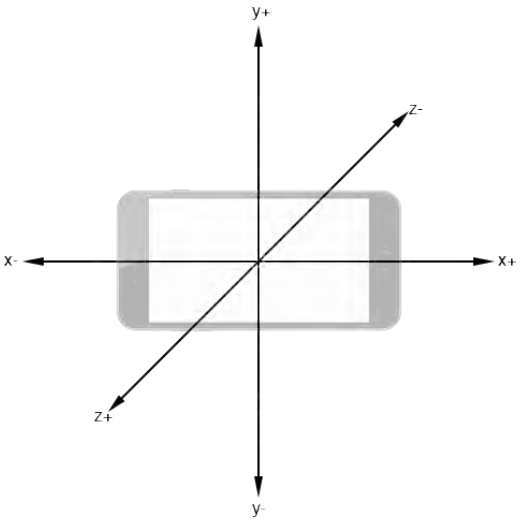


Abbildung 5.6: Koordinatensystem in der *Landscape-Left*-Orientierung zur Positionierung eines Soundobjektes

Nachdem anschließend die Lautstärke berechnet und dem Soundobjekt übergeben wurde, wird der Sound abgespielt. Wurde in den App-Einstellungen konfiguriert, dass ein Tiergeräusch häufiger als einmal abgespielt werden soll, wird ein Timer erstellt, der das Geräusch alle vier Sekunden wiederholt. Nach der Erstellung des Tierlauts wird, vor-

ausgesetzt es wurde in den Einstellungen eingestellt, ein Hintergrundgeräusch nach dem gleichen Prinzip erzeugt und parallel abgespielt. Damit bei einer Drehung des Nutzers der Sound an der richtigen Stelle abgespielt wird, wird mit Hilfe eines `CMMotionManager`[11] und der Funktion `startDeviceMotionUpdates(using:to:withHandler:)`[12] die aktuelle Position des iPhones aktualisiert. Innerhalb dieser Funktion wird dann die von Google zur Verfügung gestellte Methode `render()` aufgerufen, die wiederum die Position des Soundobjektes aktualisiert.
Klickt der Nutzer auf den angezeigten Kamerabutton finden zunächst einige grafische Änderungen statt. Der Button sowie der dunkle Hintergrund verschwinden. Dafür werden das zur Szene passende Panoramabild und, am oberen Rand des Displays, das Ergebnis angezeigt. Um das Ergebnis richtig darstellen zu können, werden zunächst alle notwendigen Daten bestimmt. Die benötigte Zeit zum Auffinden des Tieres sowie der abweichende Winkel werden berechnet und anschließend angezeigt. Der abweichende Winkel wird dabei mit der Formel 5.1 berechnet.

$$cos\alpha = \frac{\overrightarrow{a} * \overrightarrow{b}}{|\overrightarrow{a}| * \left|\overrightarrow{b}\right|} \tag{5.1}$$

Ein Beispiel zur Berechnung des Winkels zwischen der Position des Tiergeräusches und der lokalisierten Position des Nutzers ist im Folgenden zu sehen:

$$\overrightarrow{a} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \overrightarrow{b} = \begin{pmatrix} 0.3 \\ 0.7 \end{pmatrix} \tag{5.2}$$

$$cos\alpha = \frac{1 * 0.3 + 0 * 0.7}{\sqrt{1^2 + 0^2} * \sqrt{0.3^2 + 0.7^2}} = \frac{0.3}{\sqrt{1} * \sqrt{0.58}} \tag{5.3}$$

$$\alpha = \arccos(cos) = \arccos(\frac{0.3}{\sqrt{0.58}}) \approx 66,801 \tag{5.4}$$

Da die Lokalisierung nur in der horizontalen Ebene stattfindet, kann die dritte Koordinate bei der Berechnung vernachlässigt werden. Angenommen $\overrightarrow{a}$ ist die Position des Tiersounds und $\overrightarrow{b}$ ist die vom Nutzer gewählte Position (5.2). Setzt man diese beiden

[11] https://developer.apple.com/reference/coremotion/cmmotionmanager
[12] https://developer.apple.com/reference/coremotion/cmmotionmanager/1616176-startdevicemotionupdates

Vektoren in die Formel 5.1 ein und berechnet das Ergebnis (5.3), erhält man einen Winkel von ungefähr 66,8° (5.4). Das heißt der Nutzer lag um etwa 66,8° daneben.
Zusätzlich zu den berechneten Daten wird dem Nutzer über zwei Icons angezeigt, ob er innerhalb der in den Einstellungen gewählten Werte liegt oder nicht. Damit der Benutzer ein weiteres visuelles Feedback bekommt, wird ein Bild des Tieres über das 360°-Panorama-Bild gelegt. Wird die Position des Tieres korrekt erkannt, erscheint das Bild sofort beim simulierten Auslösen der Kamera. Ansonsten wird das Bild an der richtigen Position angezeigt und dem Nutzer wird die Möglichkeit geboten, sich im Panorama umzusehen und zu überprüfen wo das Tier eigentlich gewesen wäre. Die komplette Ansicht des Ergebnisses ist in Abbildung 5.7 dargestellt.

Abbildung 5.7: Angezeigtes Ergebnis mit Bild des Tieres

Damit das Tier immer an der richtigen Stelle angezeigt wird wenn sich der Nutzer im Panoramabild umsieht, muss ständig die Position des Tierbildes aktualisiert werden. Zur Berechnung der aktuellen Koordinaten wird zunächst überprüft, ob sich der Richtungsvektor des iPhones links oder rechts vom angezeigten Tier befindet. Dafür wird eine von Alecu vorgeschlagene Formel zur Berechnung eines Winkels mit Vorzeichen zwischen zwei Vektoren im 2D-Raum verwendet [2]. Mit dieser Formel wird zunächst das Kreuzprodukt und das Skalarprodukt der beiden Vektoren berechnet. Anschließend wird mit Hilfe dieser Resultate der Arkustangens (atan2), welcher zwei Argumente entgegennimmt, berechnet. Diese Funktion liefert den Winkel zwischen beiden Vektoren als Radiant. Durch das Vorzeichen dieses Ergebnisses kann festgestellt werden, ob sich der Richtungsvektor des iPhones links oder rechts der Tierposition befindet. Anschließend kann die horizontale Position des Bildes berechnet und aktualisiert werden. Da das Bild immer vertikal zentriert angezeigt wird, muss dieser Wert nicht angepasst werden. Durch

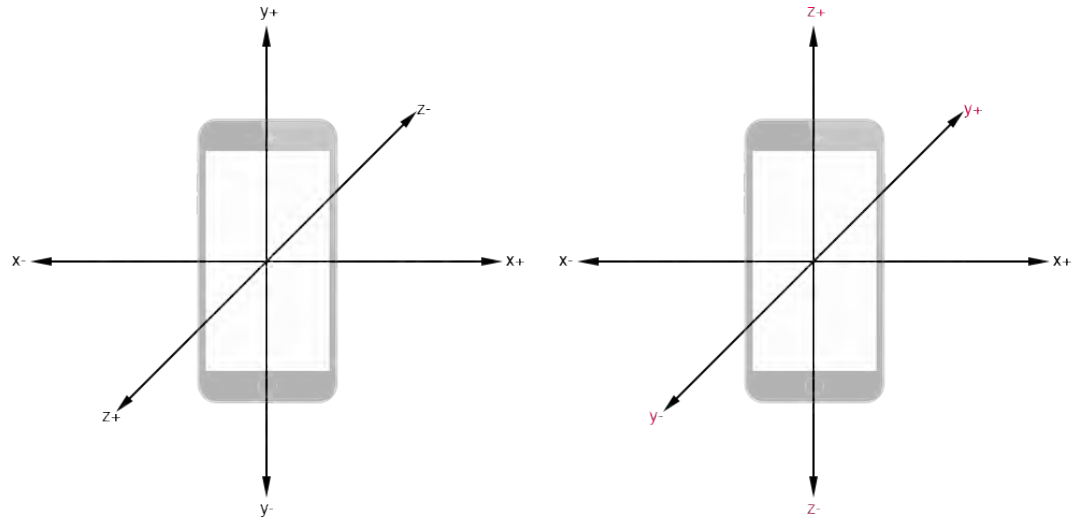
die Aktualisierung der Bildposition wird garantiert, dass der Nutzer das Tier immer an der gleichen Position sieht.
Erscheint die Ergebnisansicht zum ersten Mal innerhalb einer Szene, wird der Benutzer durch eine Meldung darauf hingewiesen, wie er zum nächsten Tier gelangt. Um auch hier eine einfache Benutzung der Anwendung zu ermöglichen, wurde dieser Schritt über die Eingabe eines Doppelklicks gelöst: Tippt der Nutzer doppelt auf das Display, wird die Szene aktualisiert. Dabei wird zunächst zufällig ein Tier der zur Szene gehörenden ausgewählt und anschließend die Position des Tieres durch Zufallswerte gesetzt. Nachdem ein Tier gewählt wurde, wird wieder, wie weiter oben bereits beschrieben, das Tiergeräusch mit Hilfe der Audio Engine geladen, positioniert und abgespielt. Der Nutzer kann schließlich erneut versuchen, die Position des Tieres zu erkennen und hat die Möglichkeit den gesamten Vorgang beliebig oft zu wiederholen.

5.2.4.4 Probleme mit dem Google Panorama View

Das Google VR SDK bietet zur Entwicklung von Virtual Reality Inhalten zwei Möglichkeiten an. Zum einen kann die Klasse `GVRPanoramaView`[13] verwendet werden. Mit dieser ist es möglich, 360°-Panorama-Bilder auf dem Smartphonedisplay anzuzeigen. Zum anderen kann mit Hilfe der Klasse `GVRVideoView`[14] ein 360°-Video auf dem Display des Endgerätes abgespielt werden. Da den Anforderungen in Kapitel 3 zufolge nach dem Auslösen der simulierten Kamera ein 360°-Panorama angezeigt werden soll, wurde bei der Entwicklung zunächst auf die Verwendung der `GVRPanoramaView`-Klasse zurückgegriffen.
In Xcode wurde zuerst über den Interface Builder aus der Objektliste ein View Controller auf den gewünschten Screen gezogen. Anschließend konnte über den Identity Inspector eine benutzerdefinierte Klasse angegeben werden. An dieser Stelle musste die Google Klasse `GVRPanoramaView` eingetragen werden. In einem weiteren Schritt wurde dann der erstellte View mit der Klasse `SceneController` verbunden. Dadurch war es möglich, problemlos ein 360°-Bild anzuzeigen, in dem sich der Nutzer umsehen kann. Allerdings traten Probleme bei der Positionierung des 3D-Sounds auf.
Zur Positionierung eines Soundobjektes müssen die drei Werte x, y und z gesetzt werden. In einem frühen Stadium der Entwicklungsphase wurde davon ausgegangen, dass sich diese Werte auf das Koordinatensystems des iPhones beziehen. Nach ersten Tests

[13] https://developers.google.com/vr/ios/reference/interface_g_v_r_panorama_view
[14] https://developers.google.com/vr/ios/reference/interface_g_v_r_video_view

wurde allerdings festgestellt, dass das Google VR SDK ein anderes Koordinatensystem verwendet. Beide Systeme sind zum Vergleich in Abbildung 5.8 dargestellt. Dabei wird vorausgesetzt, dass sich das iPhone in der *Portrait*-Orientierung befindet. Wie zu sehen ist, sind die y- und z-Achse beim zweiten System vertauscht.



(a) Koordinatensystem iPhone *Portrait*-Orientierung

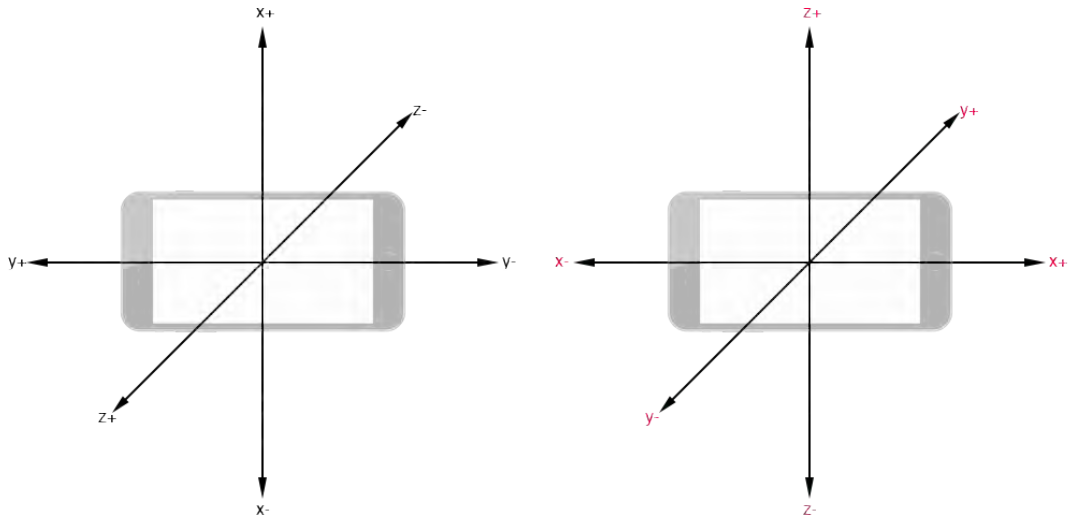
(b) Koordinatensystem Google VR SDK *Portrait*-Orientierung

Abbildung 5.8: Vergleich der beiden Koordinatensysteme

Damit das Soundobjekt bei einer Drehung des Benutzers und der damit verbundenen Drehung des Endgerätes richtig gerendert werden kann, musste sichergestellt werden, dass das gleiche Koordinatensystem verwendet wird. Um dies zu erreichen, wurden die Koordinaten zunächst auf das System des iPhones abgebildet und anschließend auf das zweite System umgerechnet. Die Positionsaktualisierung wurde anschließend mit Hilfe der Klasse `CMAttitude`[15] gelöst. Diese liefert den Standpunkt des Endgerätes zu einem bestimmten Zeitpunkt. Die gemessenen Werten können dabei zum Beispiel als Rotationsmatrix oder als Quaternion ausgelesen werden. Durch diese Werte können wiederum die Koordinaten des Soundobjektes aktualisiert werden.
Da bei der Anwendung die menschliche Lokalisationsfähigkeit nur in der horizontalen Ebene untersucht werden soll, musste bei der Umrechnung nur der gesetzte z-Wert der Google-Funktion als y-Wert übergeben und um 180° um die z-Achse rotiert werden. Bei weiteren Testdurchläufen mit unterschiedlichsten Werten konnte so eine korrekte Funk-

[15]https://developer.apple.com/reference/coremotion/cmattitude

tion des Renderings festgestellt werden. Allerdings sollte die App nach den in Kapitel 3 beschriebenen Anforderungen in der *Landscape-Left*-Orientierung ablaufen.
Bei anschließenden Tests, in denen das Smartphone in die geforderte Orientierung gedreht wurde, konnte festgestellt werden, dass sich das von Google verwendete Koordinatensystem nicht wie das iPhone-Koordinatensystem um 90° gegen den Uhrzeigersinn gedreht hat. Beide Systeme in der *Landscape-Left*-Orientierung sind wieder zum Vergleich in Abbildung 5.9 veranschaulicht.



(a) Koordinatensystem iPhone *Landscape-Left*-Orientierung

(b) Koordinatensystem Google VR SDK *Landscape-Left*-Orientierung

Abbildung 5.9: Weiterer Vergleich der beiden Koordinatensysteme

Um das Verhalten des verwendeten Koordinatensystems zu verstehen, wurde testweise das iPhone auch in die *Landscape-Right*-Orientierung gebracht und einige Tests durchgeführt. Dabei wurde festgestellt, dass das gedrehte System im Vergleich zu Abbildung 5.9 nicht um 180° um die y-Achse gedreht wurde. Ein eindeutiges Verhalten des Ausgangssystems bei einer Drehung konnte so nicht erkannt werden. Damit trotzdem eine problemlose Funktionsweise erreicht werden konnte, wurde versucht, die gesetzten Werte auf das vom iPhone verwendete auf das zweite System umzurechnen. Dadurch konnte zunächst eine korrekte Funktionsweise der Anwendung erreicht werden. Innerhalb mehrerer Tests fiel jedoch auf, dass sich die Drehung des von Google verwendeten Koordinatensystems nicht immer gleich verhält. Dadurch konnte ein richtiges Rendering des Soundobjektes nicht garantiert werden, da nicht auf einfache Weise herausgefunden werden konnte, welcher

Wert des Standpunktes des Gerätes zur Aktualisierung der jeweiligen Koordinate genutzt werden musste. Damit jedoch die gewünschte Funktionsweise der Applikation erreicht werden konnte, wurde dieser erste Ansatz verworfen und eine Lösung für das aufgetretene Problem mit der Verwendung des Google Panorama View gesucht. Die gefundene Lösung wird im folgenden Abschnitt näher beschrieben.

5.2.4.5 Lösung mit dem Google Cardboard View

Um eine Lösung für das im vorherigen Abschnitt beschriebene Problem zu finden, wurde das von Google bereitgestellte Beispiel `TreasureHunt`[16] näher betrachtet. Dabei wurde der in Objective-C geschriebene Code durchgelesen und der Ablauf des Beispiels verständlich gemacht. Dadurch konnte festgestellt werden, dass hier eine Instanz der Klasse `GVRCardboardView`[17] verwendet wurde und diese eine render-Funktion besitzt. Diese Klasse wird normalerweise zum Rendern von grafischen Elementen in Virtual Reality Anwendungen verwendet. Innerhalb der bereitgestellten Funktion wird wiederum eine Callback-Funktion aufgerufen, die als Parameter die aktuelle Transformation des iPhones übergeben bekommt. Diese übergebenen Daten können in ein Quaternion umgewandelt werden. Die verwendete Audio Engine bietet eine Funktion namens `setHeadRotation`[18] an, welche wiederum als Parameter die einzelnen Elemente eines Quaternions erwartet. So kann die aktuelle Position des Smartphones an die Audio Engine übergeben und die Position des Soundobjektes aktualisiert werden. Mit diesem Ansatz wurde schließlich versucht das in Abschnitt 5.2.4.4 geschilderte Problem beim Rendering eines Soundobjektes zu lösen.
Der zuvor verwendete Panorama View wurde dabei durch einen Cardboard View ersetzt. Außerdem wurde die Aktualisierung der Position eines Soundobjektes bei einer Rotation des iPhones nicht mehr selbst, sondern mit Hilfe der zur Verfügung gestellten Methoden gelöst. Nach einigen Tests konnte so eine zuverlässige Funktion des gewünschten Ablaufs festgestellt und das zunächst aufgetretene Problem beseitigt werden. Da allerdings der Cardboard View die Anzeige eines 360°-Panorama-Bildes nicht unterstützt, musste zusätzlich ein Panorama View hinzugefügt werden. Dieser View wurde wiederum mit Hilfe des Interface Builders zum jeweiligen Screen hinzugefügt. Dabei wurde er unterhalb des Cardboard Views positioniert. Beim Start der Untersuchung der Lokalisationsfähigkeit

[16] https://github.com/googlevr/gvr-ios-sdk/tree/master/Samples/TreasureHunt
[17] https://developers.google.com/vr/ios/reference/interface_g_v_r_cardboard_view
[18] https://developers.google.com/vr/ios/reference/interface_g_v_r_audio_engine

ist dadurch zunächst das 360°-Panorama-Bild nicht sichtbar, da es sich unterhalb des angezeigten Cardboard Views befindet. Jedoch ist es bereits vorhanden und dreht das Panorama-Bild bei einer Rotation des Smartphones mit. Klickt der Nutzer auf den angezeigten Kamera-Button wird der Cardboard View ausgeblendet und der Panorama View erscheint. Die in Kapitel 3 geforderte Funktion konnte so mit Hilfe der Verwendung von zwei unterschiedlichen Views realisiert werden.

5.2.5 Datenspeicherung

Damit das Verhalten des Nutzers später besser untersucht werden kann, wurde in Kapitel 3 ein Logging von Daten gefordert. Um das Speichern von Daten zu ermöglichen, wurde zuerst dem Einstellungen-Screen ein so genannter Switch hinzugefügt. Mit diesem kann der Nutzer das Logging der Daten aktivieren bzw. deaktivieren. Beim Starten der Überprüfung wird dann zunächst ein Logfile erstellt, in welches sämtliche notwendige Daten geschrieben werden. Um das gewünschte regelmäßige Speichern von Daten zu realisieren, wurde die von Apple bereitgestellte Klasse `Timer`[19] verwendet. Diese Klasse ermöglicht es, eine Aktion im Abstand eines angegebenen Zeitintervalls wiederholt ausführen zu lassen. Bei der Erstellung der Szene wird dafür mit Hilfe der Klassenfunktion `scheduledTimer(timeInterval:target:selector:userInfo:repeats:)`[20] ein Timer erstellt. Diesem wird unter anderem ein gewünschtes Zeitintervall in Sekunden und die auszuführende Aktion übergeben. Durch die geplante Ausführung der gewünschten Aktion wird alle 300ms die aktuelle Position des iPhones in das Logfile geschrieben. Mit der Bereitstellung dieser Daten kann so später überprüft werden, ob der Nutzer auf einer Position verharrt hat und dann das Tiergeräusch lokalisiert hat, oder ob dieser sich ständig gedreht und so die Position des Geräusches gefunden hat.
Um nicht nur die Position des iPhones, sondern auch die Leistung des Nutzers zu speichern, werden bei einem Klick auf den Kamera-Button ebenfalls Daten in das Logfile geschrieben. Hierbei werden vom erzielten Ergebnis folgende Daten gespeichert:

- gewählte Szene
- abweichender Winkel zum Tiergeräusch in Grad
- benötigte Zeit in Sekunden
- abgespieltes Tiergeräusch

[19] https://developer.apple.com/reference/foundation/timer
[20] https://developer.apple.com/reference/foundation/timer/1412416-scheduledtimer

- Koordinaten des Tiergeräusches

Damit das Ergebnis des Benutzers besser in Relation zu den gewählten Einstellungen gestellt werden kann, werden zusätzlich folgende zuvor konfigurierte Daten ebenfalls im Logfile gespeichert:

- Lautstärke des iPhones
- Lautstärke der App
- Genauigkeit in Grad
- Hintergrundgeräusch aktiviert/deaktiviert
- Anzahl der Wiederholungen des Tiergeräusches
- erlaubte Zeit zur Lokalisierung des Geräusches in Sekunden

Mit Hilfe dieser gespeicherten Daten kann das Verhalten und das erzielte Ergebnis eines Nutzers genau überprüft und bewertet werden. Ein Auszug eines Logfiles ist in Abbildung 5.10 dargestellt.

```
2017-01-25 07:51:33 +0000    1485330693474.44    device direction   x: 0.00363444 y: 0.228295 z: -0.973585
2017-01-25 07:51:33 +0000    1485330693753.25    device direction   x: 0.0394636 y: 0.254379 z: -0.966299
2017-01-25 07:51:34 +0000    1485330694054.72    device direction   x: -0.0971981 y: 0.213515 z: -0.972092
2017-01-25 07:51:34 +0000    1485330694353.36    device direction   x: -0.246801 y: 0.191116 z: -0.950033
2017-01-25 07:51:34 +0000    1485330694653.76    device direction   x: -0.378723 y: 0.181049 z: -0.907629
2017-01-25 07:51:34 +0000    1485330694953.24    device direction   x: -0.452393 y: 0.137214 z: -0.881199
2017-01-25 07:51:35 +0000    1485330695253.17    device direction   x: -0.304818 y: 0.103344 z: -0.946787
2017-01-25 07:51:35 +0000    1485330695554.92    device direction   x: -0.164707 y: 0.104069 z: -0.980837
2017-01-25 07:51:35 +0000    1485330695853.35    device direction   x: -0.10054 y: 0.12061 z: -0.987596
2017-01-25 07:51:36 +0000    1485330696152.62    device direction   x: -0.0856452 y: 0.121083 z: -0.988941
2017-01-25 07:51:36 +0000    1485330696452.98    device direction   x: -0.0392965 y: 0.133501 z: -0.990269
2017-01-25 07:51:36 +0000    1485330696723.31    RESULT: scene: 0 angle: 88.7059° time: 3.51394701004028s animal: pig animal position: x=1.0 y=0.0 z=0.0
2017-01-25 07:51:36 +0000    1485330696726.48    SETTINGS: deviceVolume: 0.125 volume: 2 accuracy: 20 background noise: 0 animal noise: 4 allowed time: 5s
```

Abbildung 5.10: Gespeicherte Daten innerhalb eines Logfiles

5.2.6 Teilen-Funktion

Eine weitere Anforderung an die iOS Anwendung war die Möglichkeit, gespeicherte Daten per E-Mail zu versenden. Um diese Anforderung zu realisieren, wurde die von iOS bekannte Teilen-Funktion implementiert. Dafür wurde zunächst über den Interface Builder ein Button zum bereits existierenden Einstellungen-Screen hinzugefügt und anschließend mit der Klasse `SettingsController` verbunden. Bei einem Klick auf den Button wird zunächst der Name und dadurch der Pfad der gespeicherten Datei

ausgelesen. Der Pfad wird anschließend zu einer `NSURL`[21] umgewandelt. Eine `NSURL` kann unter anderem den Pfad einer lokal gespeicherten Datei beinhalten, der im nächsten Schritt benötigt wird. Als nächstes wird ein `UIActivityViewController`[22] Objekt erstellt. Dieses erhält als Übergabewert unter anderem die zuvor erzeugte `NSURL`, welche den Pfad zu der zu teilenden Datei enthält. Mit Hilfe eines `UIActivityViewController` können Daten in Social-Media-Kanälen geteilt, per SMS oder E-Mail versendet oder auch nur kopiert werden. Außerdem können Apps eigene Funktionen mit Hilfe dieses Controllers zur Verfügung stellen. Das erzeugte Objekt wird anschließend dem Nutzer angezeigt. Dieser kann, wie in Abbildung 5.11 zu sehen ist, nun auswählen, dass er die Datei per E-Mail versenden will. Dabei wird die iOS E-Mail Anwendung geöffnet und durch Eingabe einer Zieladresse können die gespeicherten Daten problemlos auf einen Computer übertragen werden.

Abbildung 5.11: Ansicht der Teilen-Funktion der Anwendung

[21] https://developer.apple.com/reference/foundation/nsurl
[22] https://developer.apple.com/reference/uikit/uiactivityviewcontroller

5.3 Testen

Ein wichtiges Element während der Entwicklung einer Anwendung ist das Testen. Nachdem die App in der Entwicklungsphase immer wieder überprüft wurde, wurde sie zusätzlich von vier weiteren Personen getestet. Dabei sollte vor allem herausgefunden werden, ob eine reibungslose Untersuchung der menschlichen Lokalisationsfähigkeit möglich ist und ob die App einfach und selbsterklärend zu bedienen ist. Den Testern wurden dabei keine weiteren Angaben gemacht und es wurden auch keine expliziten Aufgaben gestellt. Ein durch die Tester gefundenes Problem war, dass bei einem Wechsel der Orientierung, zum Beispiel aus der *Landscape-Left-* in die *Portrait*-Orientierung und wieder zurück, erneut eine Szene erstellt wurde. Dies hatte zur Folge, dass mehrere Tiergeräusche durcheinander zur selben Zeit abgespielt wurden und so eine korrekte Lokalisierung unmöglich war. Dieses Problem wurde mit einer Abfrage, die überprüft, ob bereits eine Szene erstellt wurde, gelöst. Anschließend konnte das Testen der Lokalisierung trotz Änderung der Orientierung problemlos durchgeführt werden.
Ein weiteres Problem lag darin, dass bis zum Abspielen eines Sounds zu lange gewartet werden musste. Um eine Lösung für diese Problematik zu finden wurde versucht den Code zu optimieren. Allerdings kann beim ersten Auftreten eines Sounds kein schnelleres Abspielen erreicht werden, da die jeweilige Sounddatei zunächst geladen werden muss und erst anschließend abgespielt werden kann. Jedoch war eine Optimierung bei der Wiederholung eines Tiergeräusches möglich. Hier wurde die Datei zunächst jedes Mal nochmals neu geladen und anschließend wieder gelöscht. Da es aber reicht, die Datei nur vor dem ersten Abspielen zu laden und nach dem letzten Mal wieder zu löschen, konnte der Code an dieser Stelle optimiert werden. Dadurch konnte ein zu langes Warten auf das Tiergeräusch verhindert werden.
Zusätzlich zu den funktionalen Schwierigkeiten wurden von den Testern auch noch grafische Verbesserungen vorgeschlagen. Unter anderem wurde bemängelt, dass der Button zum Zurückkehren in das Menü schlecht erreichbar ist und zu nah am Displayrand positioniert wurde. Dieser Button war zunächst am oberen linken Rand des Displays positioniert. Um ein leichteres Erreichen des Buttons zu ermöglichen, wurde er etwas vergrößert und am oberen rechten Rand des Displays, weiter vom Rand entfernt, positioniert. Desweiteren wurde der auf einem Panorama-Bild von Google automatisch angezeigte Info-Button bemängelt. Nach einer Recherche wurde jedoch festgestellt, dass es nicht möglich ist, diesen auszublenden und er angezeigt wird, um zur Hilfeseite von Google Cardboard zu kommen.

Ein weiteres grafisches Problem lag darin, dass die angezeigte Warnung vor dem Start einer Szene, vorausgesetzt das iPhone wird in der *Portrait*-Orientierung gehalten, bei einer Drehung in die *Landscape-Right*-Orientierung nicht richtig angezeigt wurde. Dieser Anzeigefehler wurde gelöst, indem jetzt eine Rotation des Endgerätes in die *Landscape-Right*-Orientierung erkannt wird. Ist dies der Fall, wird die Position der angezeigten Warnung neu berechnet, aktualisiert und anschließend richtig dargestellt.
Durch die aufgefundenen Probleme der Tester konnte durch die jeweilgen Lösungswege die Funktion und Bedienbarkeit der Anwendung nochmals verbessert werden. Eine problemlose Bedienung und eine korrekte Funktionsweise ist nun für jeden Nutzer möglich.

6

Anforderungsabgleich

In diesem Kapitel werden die Anforderungen aus Kapitel 3 mit den Eigenschaften der entwickelten iOS App verglichen. Jede einzelne Anforderung so wie die Antwort auf die Frage, ob diese umgesetzt wurde, ist in Tabelle 6.1 zusammengefasst.
In der Tabelle ist schnell zu sehen, dass alle geforderten funktionalen Anforderungen umgesetzt wurden. Die gewünschte Funktion der Anwendung wurde implementiert. Zusätzlich wurde ein visuelles Feedback für den Nutzer hinzugefügt. Dadurch kann dieser sofort sein erzieltes Ergebnis, darunter die benötigte Zeit zum Auffinden eines Tiergeräusches sowie der abweichende Winkel, anschauen und so seine Leistung richtig einschätzen. Die gewünschten Konfigurationsmöglichkeiten der Anwendung wurden ebenfalls umgesetzt. Hier wurde ein weiterer Einstellungsparameter hinzugefügt. Durch diesen kann eingestellt werden, wie häufig ein Tiergeräusch abgespielt werden soll. Dadurch kann die Untersuchung der Lokalisiationsfähigkeit nochmals schwerer bzw. leichter gemacht werden. Umgesetzt wurde auch das geforderte Daten-Logging. Bei der finalen Lösung wird alle 300ms die aktuelle Position des Smartphones gespeichert. Außerdem werden bei einem Klick auf den Auslösebutton das Ergebnis des Nutzers sowie alle gewählten Einstellungen gespeichert. Wird die Untersuchung beendet und zum Menü zurückgekehrt, kann unter den Einstellungen die Möglichkeit zum Teilen der aufgezeichneten Daten gefunden werden. Mit der Bereitstellung dieser geforderten Funktion kann schnell und einfach die erstellte Datei mit den gespeicherten Daten per E-Mail auf einen Computer übertragen werden. Anschließend kann dort das Verhalten und die Leistung des Nutzers näher untersucht und genaustens analysiert werden.
Ebenfalls wurden die nicht-funktionalen Anforderungen, darunter die einfache Steuerung der App insbesondere während der Untersuchung und die Verfügbarkeit der App für alle iPhone und iPad Modelle erfolgreich umgesetzt. Lediglich die Anforderung zum schnellen Laden eines Sounds konnte nicht hunderprozentig umgesetzt werden. Hier kommt es vor allem auf die Leistung des verwendeten Endgerätes an. Auf neueren Modellen

wurden Sounds recht zügig geladen wohingegen der Vorgang auf älteren Modellen etwas länger dauerte. Da aber die jeweiligen Dateien mit Hilfe einer von Google bereitgestellten Funktion geladen werden, war eine Optimierung nicht möglich.
Nochmals zusammengefasst wurden alle funktionalen und fast alle nicht-funktionalen Anforderungen an die zu entwickelnde mobile Anwendung umgesetzt. Außerdem wurden zwei zusätzliche Anforderungen hinzugefügt, die ebenfalls implementiert wurden. Diese waren zum einen die Ergebnisanzeige als Feedback für den Nutzer und zum anderen die mögliche Einstellung der Anzahl an Wiederholungen des abzuspielenden Tiergeräusches.

Tabelle 6.1: Veranschaulichung der umgesetzten Anforderungen

Anforderung	**Gefordert?**	**Umgesetzt?**
App-Ablauf		
Menü zur Auswahl von unterschiedlichen Szenen	Ja	✓
Dunkler Bildschirm mit einem Auslöse-Button	Ja	✓
Abspielen eines Tiergeräusches aus einer zufälligen Richtung	Ja	✓
Darstellung eines 360°-Panorama-Bildes	Ja	✓
Darstellung des jeweligen Tieres	Ja	✓
Nutzer kann sich im Panorama-Bild umschauen	Ja	✓
Nutzer kann den Ablauf wiederholen	Ja	✓
Einfache Steuerung	Ja	✓
Sounds sollen schnell genug geladen werden	Ja	-
Ergebnisanzeige als Feedback für den Nutzer	Nein	✓
App-Einstellungen		
Konfiguration des App-Ablaufs	Ja	✓
Lautstärke	Ja	✓
Genauigkeit	Ja	✓
Hintergrundgeräusch	Ja	✓
Erlaubte Zeit	Ja	✓
Anzahl Tiergeräusche	Nein	✓
Daten-Logging		
Speichern der Position des Smartphones alle x Millisekunden	Ja	✓

Speichern des Ergebnises bei Klick auf den Kamera-Button	Ja	✓
Speichern der gewählten Einstellungen bei Klick auf den Kamera-Button	Ja	✓
Teilen-Funktion		
Möglichkeit zur Übertragung der gespeicherten Datei per E-Mail	Ja	✓
Sonstiges		
Verfügbarkeit für alle iPhone und iPad Modelle	Ja	✓

7 Evaluierung

Nachdem die Entwicklung der Anwendung abgeschlossen war, wurde eine Evaluierung der App durchgeführt. Innerhalb dieses Kapitels wird zunächst die Planung der durchgeführten Studie näher beschrieben. Da die erzielten Ergebnisse mit den Resultaten vorheriger Studien, in denen andere Anwendungen verwendet wurden, verglichen werden sollten, mussten einige Anpassungen an der App vorgenommen werden. Diese sind in Abschnitt 7.2 erklärt. Die Durchführung der Studie wird schließlich in Abschnitt 7.3 näher erläutert. Bevor im letzten Teil die Ergebnisse diskutiert und mit den vorherigen Resultaten verglichen werden, wird zunächst die Auswertung der durchgeführten Studie geschildert.

7.1 Planung der Studie

Wir bereits erwähnt, sollten die Ergebnisse dieser Studie mit denen von vorhergegangenen Studien aus Abschlussarbeiten von Weidhaas und Henkel verglichen werden [53][26]. Das heißt, die Studie sollte weitestgehend gleich aufgebaut sein, damit ein problemloser Vergleich der Resultate möglich ist. Es sollte also zum einen die richtige Funktionsweise der entwickelten App und zum anderen der Einfluss durch die Anzahl von zu hörenden Geräuschen auf die erzielte Genauigkeit und Geschwindigkeit überprüft werden. Genauer gesagt sollten durch die Durchführung der Studie Daten von erzielten Winkeln und dafür benötigten Zeiten gesammelt werden, damit ein späterer Vergleich der unterschiedlichen Ergebnisse überhaupt möglich ist.

7.1.1 Hypothesen

Während der Planung wurden folgende vier Hypothesen aufgestellt, welche innerhalb der Studie validiert werden sollten:

- Der erzielte abweichende Winkel bei der Lokalisierung eines Tierlauts ist abhängig von der Anzahl der zu hörenden Geräusche.
- Die benötigte Zeit bei der Lokalisierung eines Tierlauts ist abhängig von der Anzahl der zu hörenden Geräusche.
- Mit der entwickelten App wird ein geringerer durchschnittlicher Winkel erzielt als mit vorherigen Anwendungen.
- Mit der entwickelten App wird eine geringere durchschnittliche benötigte Zeit erzielt als mit vorherigen Anwendungen.

7.1.2 Variablen

Damit die Hypothesen validiert werden konnten, mussten zunächst einige Variablen definiert werden. Dazu wurden zum einen unabhängige Variablen und zum anderen abhängige Variablen definiert. Unabhängige Variablen geben eine Ursache an, die durch das Design der Studie verändert wird. Sie werden als unabhängig bezeichnet, da sie unabhängig für das Verhalten eines Teilnehmers sind [17]. Folgende unabhängige Variablen wurden definiert:

- Alter
- Geschlecht
- Erkankungen
- Erfahrung mit mobilen Geräten
- Smartphone, das normalerweise verwendet wird
- Art der verwendeten Kopfhörer
- Erfahrung mit Videospielen
- Anzahl an zu hörenden Tiergeräuschen
- Hintergrundgeräusch

Abhängige Variablen sind dagegen Eigenschaften, die innerhalb eines Experiments gemessen werden. Sie sind abhängig von Änderungen der unabhängigen Variablen und müssen messbar sein [17]. Folgende abhängige Variablen wurden definiert:

- Winkel in Grad

- Benötigte Zeit in Sekunden
- Position des Geräts während der Untersuchung als Vektor (x,y,z)

7.1.3 Durchgänge

Damit die aufgestellten Hypothesen überprüft werden konnten, musste jeder Teilnehmer der Studie drei Durchgänge mit unterschiedlichen Einstellungen durchführen. Diese drei Durchgänge waren:

(A) ein Tierlaut ohne Hintergrundgeräusch

(B) zwei Tierlaute ohne Hintergrundgeräusch

(C) zwei Tierlaute mit Hintergrundgeräusch

Innerhalb jedes Durchgangs musste je dreimal die Position eines bzw. zweier Tierlautes/e erkannt werden. Da jeder einzelne Teilnehmer jeden Durchgang durchführen musste, war es möglich, dass innerhalb des ersten ausgeführten Durchgangs Erfahrungen gewonnen werden konnten, welche sich wiederum positiv auf die weiteren zwei Folgenden auswirken konnten [17]. Obwohl es unmöglich ist, diese Auswirkungen vollständig zu eliminieren, wurde die Reihenfolge der drei Durchgänge variiert. Die daraus folgende Kontrolle eines möglichen Lerneffektes wird auch *counterbalancing* genannt. Um unterschiedliche Reihenfolgen zu bekommen, wurde das in Tabelle 7.1 dargestellte Latin Square aufgestellt.

Tabelle 7.1: Latin Square

A	C	B
B	A	C
C	B	A

Mit Hilfe dieses Latin Squares wurden folgende sechs unterschiedliche Reihenfolgen erstellt:

- A, B, C
- A, C, B
- B, A, C
- B, C, A
- C, A, B
- C, B, A

7.2 Anpassung der App

Damit die Studie korrekt durchgeführt werden konnte, mussten zunächst einige Anpassungen an der entwickelten App vorgenommen werden. Bisher war es nicht möglich, in den Einstellungen auszuwählen, dass ein Tierlaut unendlich oft wiederholt werden sollte. Ebenfalls musste bisher immer ein Zeitlimit zum Auffinden der Position eines Sounds gewählt werden. Um den Studienteilnehmern diese fehlenden Funktionen zu bieten und damit den selben Studienablauf wie bei früheren Studien zu ermöglichen, wurde den Schiebereglern ein weiterer Wert hinzugefügt. Die Anzahl der Wiederholungen eines Tiergeräusches kann so maximal auf unendlich gestellt werden. Ebenfalls gibt es bei der Einstellung der erlaubten Zeit nun die Möglichkeit, eine unbegrenzte Zeitspanne zu wählen.
Außerdem gab es bisher noch keine Option, dass zwei Tiergeräusche gleichzeitig abgespielt werden und deren Position gefunden werden musste. Um dies zu ermöglichen und dadurch den identischen Studienablauf zu garantieren, wurde zunächst ein weiterer Regler dem Einstellungsscreen hinzugefügt. Mit diesem kann nun das gleichzeitige Abspielen eines zweiten Tierlauts aktiviert werden. Dabei wird der zweite Tierlaut wie bereits in Abschnitt 5.2.4.3 beschrieben erstellt und abgespielt. Damit der Studienteilnehmer weiß, welche Position er zuerst erkennen muss, wird ihm während der Untersuchung seiner Lokalisationsfähigkeit eine Meldung angezeigt. Diese gibt an welcher Tierlaut gefunden werden muss. Die entwickelten Anpassungen, welche visuell vor allem den Einstellungen-Screen betreffen, sind in Abbildung 7.1 dargestellt.

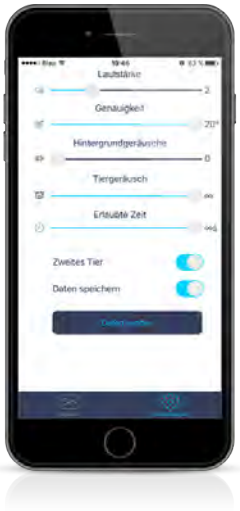


Abbildung 7.1: Einstellungen-Screen mit den entwickelten Erweiterungen

7.3 Durchführung der Studie

Zur Durchführung der Studie bekam jeder Teilnehmer zwei E-Mails zugesendet. Zunächst erhielt jeder eine Einladung von Apple TestFlight zum Downloaden der entwickelten App. TestFlight bietet die Möglichkeit, bis zu 2000 Personen zum Testen einer App für iOS, watchOS oder tvOS einzuladen, bevor diese im offiziellen App Store veröffentlicht wird [11]. Mit der zweiten E-Mail erhielt jeder Teilnehmer eine Anleitung zur Durchführung der Studie. Eine dieser Anleitungen ist im Anhang B zu sehen. Sie beinhaltet einen kurzen Einleitungstext, einen Fragebogen zur Erfassung von demographischen Daten und die durchzuführenden Durchgänge in einer der zuvor erstellten Reihenfolge.
Jeder Teilnehmer musste zuerst die App auf seinem iPhone installieren und anschließend den Fragebogen beantworten. Die Antworten, unter anderem zu Alter, Geschlecht, Gehörerkrankungen und Erfahrung mit mobilen Geräten, wurden dabei wiederum per E-Mail verschickt. Anschließend musste jeder Proband alle drei Durchgänge durcharbeiten. Am Ende jedes Durchgangs mussten die aufgezeichneten Daten per Mail versendet werden. Die komplette Teilnahme der Studie dauerte etwa 20 Minuten.

7.4 Auswertung

An der Studie nahmen insgesamt 24 Teilnehmer (w: 12, m: 12) im Alter zwischen 20 und 56 Jahren (M: 28.29, SD: 9.438), welche vor allem im Freundes- und Bekanntenkreis gesucht wurden, teil. In den folgenden zwei Unterabschnitten wird die Auswertung des Fragebogens und der drei unterschiedlichen Durchgänge näher erläutert.

7.4.1 Fragebogen

Wie in Abbildung 7.2 zu sehen ist liegt die durchschnittliche Erfahrung mit mobilen Geräten insgesamt bei 3,79 von fünf möglichen Punkten. Die männlichen Teilnehmer (4,17) hatten im Schnitt mehr Erfahrung im Vergleich zu den weiblichen Teilnehmer (3,42). Da allerdings die gesamte Erfahrung im oberen Drittel der Skala lag, kann davon ausgegangen werden, dass eine problemlose Nutzung der App möglich war.

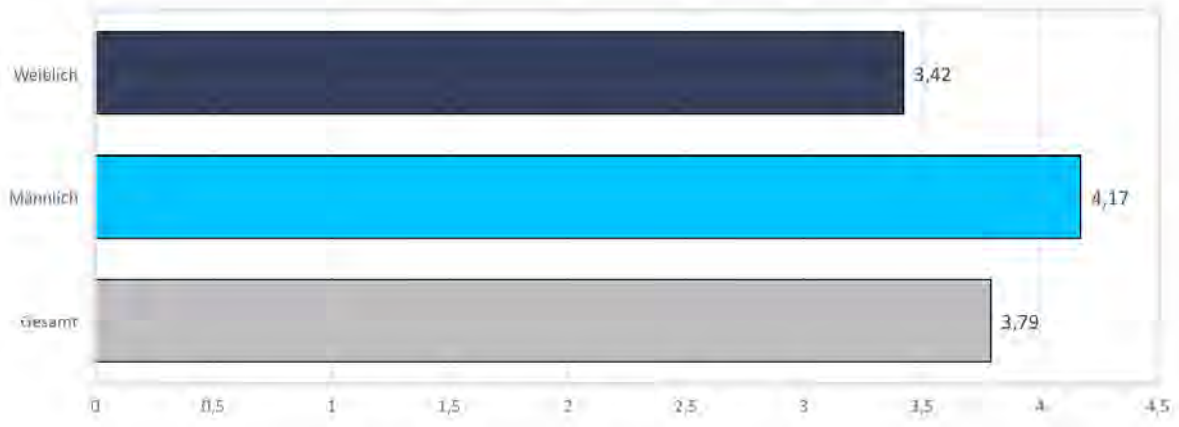


Abbildung 7.2: Durchschnittserfahrung mit mobilen Geräten

In Abbildung 7.3 ist die durchschnittliche Erfahrung der Teilnehmer mit Videospielen zu sehen. Auch hier fällt auf, dass die männlichen Teilnehmer (3,5) 1,75 Skalenpunkte mehr Erfahrung hatten als die weiblichen Teilnehmer (1,75). Insgesamt lag der durchschnittliche Erfahrungswert mit Videospielen bei 2,63 von fünf möglichen Punkten.

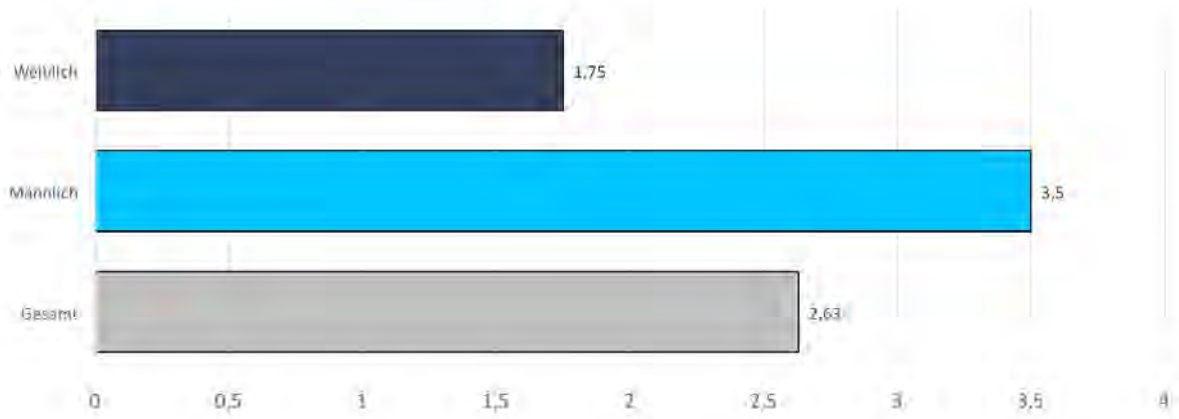


Abbildung 7.3: Durchschnittserfahrung mit Videospielen

Da die verwendete Art von Kopfhörern ebenfalls Einfluss auf den erzielten Winkel und die benötigte Zeit bei der Lokalisierung eines Geräusches haben konnte, wurde innerhalb des Fragebogens nach der verwendeten Art gefragt. Wie in Abbildung 7.4 zu sehen ist, wurden jeweiles zwölf On-Ear und In-Ear Kopfhörer von den Teilnehmern während der Studie verwendet.

Abbildung 7.4: Anzahl verwendeter Kopfhörer

Ebenfalls kann das üblicherweise verwendete Betriebssystem Einfluss auf die Bedienung der App und dadurch auch auf die erzielten Werte haben. Deshalb wurde innerhalb des Fragebogens nach dem normalerweise genutzten Smartphone gefragt. Da viele unterschiedliche Geräte genannt wurden, war eine Auswertung nach der Anzahl des für gewöhnlich verwendeten Betriebssystems sinnvoller. Wie in Abbildung 7.5 zu sehen ist nutzen 17 Personen ein iPhone mit iOS und 7 Personen ein Android Endgerät.

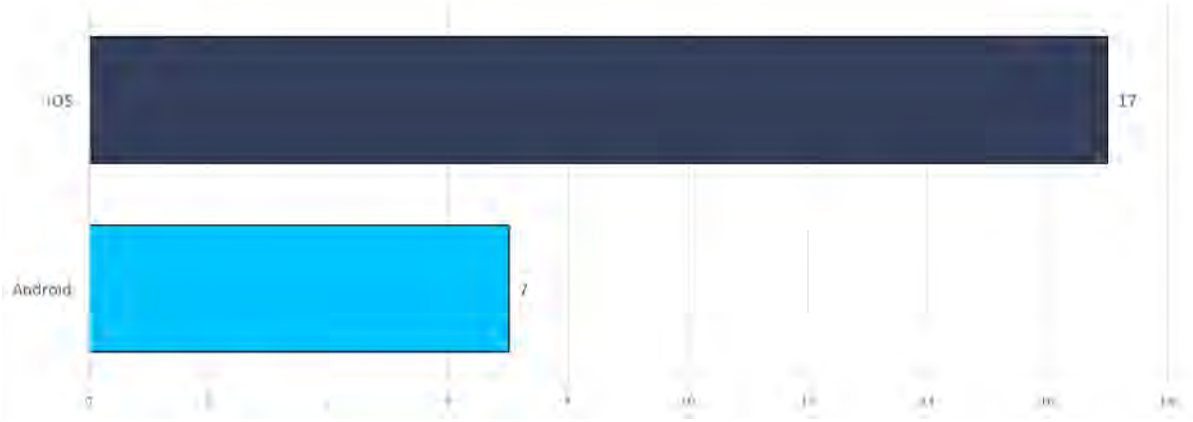


Abbildung 7.5: Anzahl der üblicherweise verwendeten Betriebssysteme

7.4.2 Durchgänge

Damit die in Abschnitt 7.1.1 aufgestellten Hypothesen für gültig erklärt oder abgelehnt werden konnten, wurden mit Hilfe der aufgezeichneten Daten Mittelwertberechnungen und vor allem Varianzanalysen durchgeführt. Mit Hilfe einer Varianzanalyse (ANOVA)

können Gegebenheiten analysiert werden, in welchen zwei oder mehr Bedingungen miteinander verglichen werden sollen [17]. Alle durchgeführten Analysen wurden mit Hilfe von IBM SPSS Statistics Version 24.0.0. angefertigt. Das Signifikanzniveau lag bei allen Auswertungen bei 5 % ($p < .05$).

Erzielte Winkel

Damit eine Varianzanalyse zur Untersuchung des erzielten Winkels in Abhängigkeit der Anzahl von zu hörenden Geräuschen durchgeführt werden konnte, wurden zunächst alle aufgezeichneten Daten in eine tabellarische Form gebracht. Dabei wurden die Bezeichnungen der einzelnen Vorgänge A, B und C mit den Werten 1, 2 und 3 ersetzt. Das Ergebnis der schließlich durchgeführten Varianzanalyse ist in folgender Tabelle zu sehen.

Tabelle 7.2: Ergebnis der Varianzanalyse zur Untersuchung der Winkel

	Quadrat-summe	**df**	**Mittel der Quadrate**	**F**	**Signifikanz**
Zwischen den Gruppen	3074,209	2	1537,105	1,287	,277
Innerhalb der Gruppen	426325,039	357	1194,188		
Gesamt	429399,248	359			

Um den durchschnittlichen Winkel mit den Ergebnissen von anderen Anwendungen vergleichen zu können, wurde zusätzlich der Mittelwerte aller erzielten Winkel berechnet. Dieses Ergebnis ist in Tabelle 7.3 zu sehen.

Tabelle 7.3: Durchschnittlicher Winkel

Mittelwert	**N**	**Standardabweichung**
23,0042147	360	34,5846524

Benötigte Zeit

Bei der Durchführung der Varianzanalyse zur Untersuchung der benötigten Zeit in Abhängigkeit von der Anzahl der zu hörenden Geräuschen wurden zunächst ebenfalls alle aufgezeichneten Daten in eine tabelarische Form gebracht. Dabei wurden, wie bereits

bei der Varianzanalyse des Winkels, ebenfalls die Bezeichnungen der einzelnen Durchgänge mit passenden Werten ersetzt. Zusätzlich mussten die benötigten Zeiten bei den Durchgängen mit zwei Tieren berechnet werden. Hier wurde nur die insgesamt benötigte Zeit, zum Auffinden von beiden Tieren, aufgezeichnet. Das heißt vom Zeitstempel der Erkennung des zweiten Tieres musste der Zeitstempel der Erkennung des ersten Tieres subtrahiert werden. Anschließend konnte mit den tabellarisch gespeicherten Daten die Varianzanalyse durchgeführt werden. Die Ergebnisse sind in Tabelle 7.4 zu sehen.

Tabelle 7.4: Ergebnis der Varianzanalyse zur Untersuchung der benötigten Zeit

	Quadrat-summe	**df**	**Mittel der Quadrate**	**F**	**Signifikanz**
Zwischen den Gruppen	599,134	2	299,567	7,760	,001
Innerhalb der Gruppen	13782,226	357	38,606		
Gesamt	14381,360	359			

Der Mittelwert der benötigten Zeit wurde ebenfalls berechnet, um einen Vergleich zu den Ergebnissen, die mit anderen Anwendungen erzielt wurden, herstellen zu können. Die durchschnittlich benötigte Zeit zum Auffinden eines Geräusches ist in Tabelle 7.5 dargestellt.

Tabelle 7.5: Durchschnittliche benötigte Zeit

Mittelwert	**N**	**Standardabweichung**
9,980462	360	6,3292574

Art der Kopfhörer

Um den Einfluss von On-Ear- bzw. In-Ear-Kopfhörern überprüfen zu können, wurden sowohl der durchschnittlich erzielte Winkel als auch die durchschnittlich benötigte Zeit in Abhängigkeit der verwendeten Kopfhörer ausgewertet. Damit diese Werte berechnet werden konnten, wurde zu den, bereits in Form einer Tabelle angeordneten, Daten eine weitere Spalte mit der jeweiligen Art der verwendeten Kopfhörer hinzugefügt. Die Ergebnisse sind in folgender Tabelle zu sehen.

Tabelle 7.6: Erzielte Leistungen in Abhängigkeit der verwendeten Kopfhörer

Kopfhörer	Bezeichnung	Winkel	Zeit
In-Ear	Mittelwert	22,1665053	9,987335
	N	180	180
	Standardabweichung	36,0953194	5,5591714
On-Ear	Mittelwert	23,8419241	9,973588
	N	180	180
	Standardabweichung	34,5846524	7,0312384

Üblicherweise verwendetes Betriebssystem

Da auch das üblicherweise benutzte Betriebssystem einen Einfluss auf die erreichten Ergebnisse haben konnte, wurden ebenfalls die durchschnittlich erzielten Werte in Abhängigkeit des normalerweise verwendeten Betriebssystems berechnet. Dafür wurde nochmals eine weitere Spalte mit der Angabe des Betriebssystems zu den gespeicherten Daten hinzugefügt. Die anschließend berechneten Werte sind in Tabelle 7.7 angeführt.

Tabelle 7.7: Erzielte Leistungen in Abhängigkeit des üblicherweise verwendeten Betriebssystems

Betriebssystem	Bezeichnung	Winkel	Zeit
Android	Mittelwert	28,5281178	7,845062
	N	120	120
	Standardabweichung	39,9210932	4,0729948
iOS	Mittelwert	20,2422632	11,048162
	N	240	240
	Standardabweichung	31,3074422	6,9626761

7.5 Diskussion der Ergebnisse

Das Ergebnis der Varianzanalyse zur Untersuchung des erzielten Winkels in Abhängigkeit der Anzahl an zu hörenden Geräuschen zeigt eine Signifikanz von .277. Da das verwendete Signifikanzniveau bei .05 liegt hat die Anzahl an zu hörenden Geräuschen keinen signifikanten Einfluss auf den erzielten Winkel. Die erste in Abschnitt 7.1.1 aufgestell-

te Hypothese kann daher nicht bestätigt werden. Vergleicht man den durchschnittlichen Winkel mit den ausgewerteten Ergebnissen von Weidhaas ist zu sehen, dass die Differenz größer als bei einer älteren iOS Anwendung ist. Das heißt, die dritte aufgestellte Hypothese kann ebenfalls nicht für gültig erklärt werden. In Tabelle 7.8 ist der innerhalb dieser Arbeit ermittelte durchschnittliche Winkel im Vergleich zu früheren Auswertungen zu sehen.

Tabelle 7.8: Durchschnittlich erzielter Winkel im Vergleich zu ermittelten Ergebnissen von Weidhaas [53]

	Aktuelle Auswertung	**Ältere Auswertungen**			
Plattform	iOS	iOS	Android	Windows	Web
Winkel	23,0	13,3	19,6	21,8	55,5

Das schlechtere Ergebnis kann zum einen damit zusammenhängen, dass das verwendete Google SDK zu ungenau ist. Dies ist allerdings eher unwahrscheinlich. Zum anderen kann es mit der Art der Druchführung der Studie zusammenhängen. Alle Teilnehmer haben die Studie alleine und ohne Aufsicht zu Hause ausgeführt. Dadurch ist es möglich, dass viele die einzelnen Durchgänge nicht konzentriert genug durchgeführt haben und vielleicht durch andere Umstände und Geräusche abgelenkt wurden. Außerdem kann es auch sein, dass die Teilnehmer sich nicht genug Zeit genommen haben und die Studie recht schnell durchführen wollten. Daher wurde nicht genau gehört von wo das zu findende Geräusch genau kommt und schnell eine Vermutung getroffen.
Wie in Tabelle 7.4 zu sehen ist, erzielte die Varianzanalyse zur Untersuchung der benötigten Zeit in Abhängigkeit der Anzahl an zu hörenden Geräuschen eine Signifikanz von .001. Das heißt die Anzahl an zu hörenden Tierlauten hat einen signifikanten Einfluss auf die benötigte Zeit zum Auffinden der Position eines Sounds. Die zweite aufgestellte Hypothese ist somit gültig. Vergleicht man auch hier die durchschnittlich benötigte Zeit mit den Ergebnissen von Weidhaas, ist zu sehen, dass mit beiden iOS Anwendungen eine ziemlich ähnliche Zeit erreicht wurde. Da aber die erzielte Durchschnittszeit minimal schlechter ist, kann die vierte Hypothese nicht bestätigt werden. Die ermittelte Durchschnittszeit ist in Tabelle 7.9 im Vergleich zu früheren Auswertungen veranschaulicht.

Tabelle 7.9: Erzielte benötigte Zeit im Vergleich zu ermittelten Ergebnissen von Weidhaas [53]

	Aktuelle Auswertung	**Ältere Auswertungen**			
Plattform	iOS	iOS	Android	Windows	Web
Benötigte Zeit	9,98	9,74	12,12	15,97	14,50

Wie zu sehen ist, wurde mit der entwickelten Anwendung die zweitbeste benötigte Durchschnittszeit erzielt. Dies könnte allerdings ein weiterer Hinweis dafür sein, dass die Teilnehmer die Studie zu schnell und unkonzentriert durchgeführt haben. Wie bereits erwähnt wurde, könnte darunter die Genauigkeit des Winkels gelitten haben.
Um die ausgewerteten Daten besser zu veranschaulichen, wurden Punktdiagramme erstellt. Die drei folgenden Abbildungen zeigen jeweils ein Punktdiagramm für jeden einzelnen durchgeführten Durchgang der Studie.

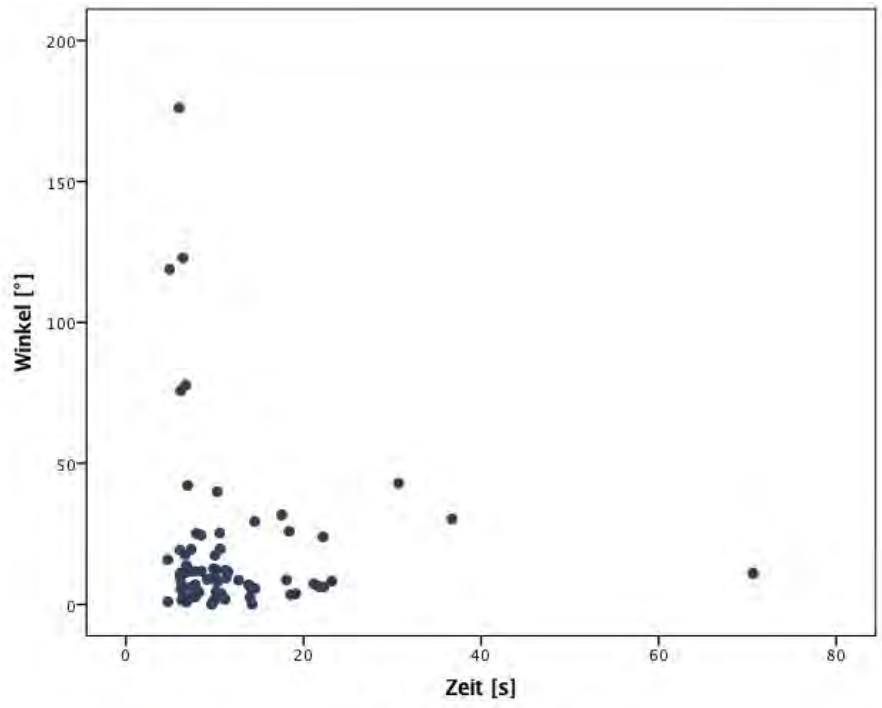


Abbildung 7.6: Punktdiagramm der Auswertung des ersten Durchgangs

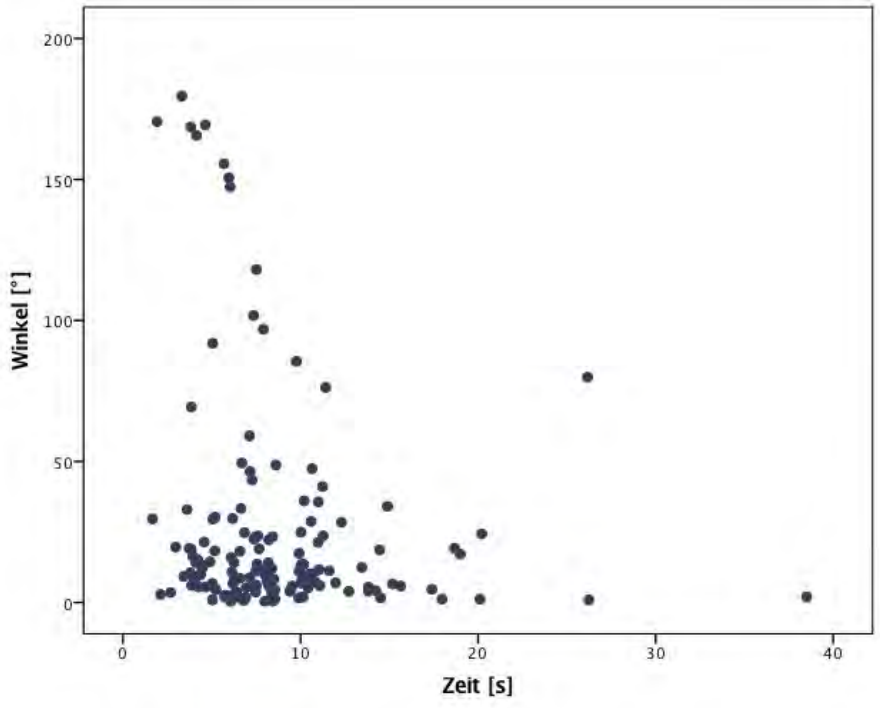


Abbildung 7.7: Punktdiagramm der Auswertung des zweiten Durchgangs

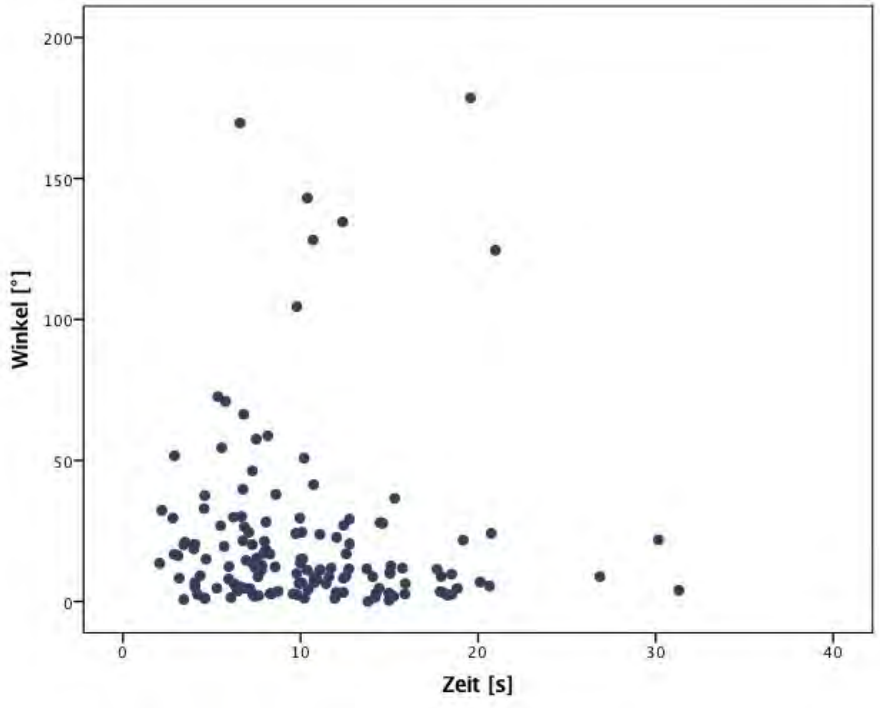


Abbildung 7.8: Punktdiagramm der Auswertung des dritten Durchgangs

Aus den drei dargestellten Diagrammen lassen sich alle erzielten Winkel und benötigten Zeiten ablesen. Es ist gut zu erkennen, dass die benötigte Zeit in Durchgang zwei und drei im Vergleich zum ersten Durchgang verteilter ist. Genauer gesagt befinden sich mehr der aufgezeichneten Daten der letzten zwei Durchgänge zwischen ungefähr 10 und 20 Sekunden. Gut zu erkennen ist ebenfalls, dass sich der Großteil der erzielten Winkeldifferenzen in allen drei Durchgängen in einem Intervall zwischen ungefähr 0° und 50° befinden. Allerdings gibt es auch Werte, die stark von diesem Intervall abweichen. Dabei fällt auf, dass viele dieser Werte in der Nähe von 180° liegen. Daraus lässt sich ableiten, dass einige Teilnehmer nicht genau erkannten, ob ein Geräusch vor oder hinter ihnen lag.
Wie der Auswertung im vorherigen Abschnitt entnommen werden kann, hat die Art der verwendeten Kopfhörer keinen sehr großen Einfluss auf die erzielten Leistungen der Teilnehmer. Vermutet wurde allerdings, dass mit On-Ear-Kopfhörern ein besseres Ergebnis erzielt werden kann. Die berechneten Ergebnisse zeigen aber eher das Gegenteil. Die benötigte Durchschnittszeit ist bei beiden Kopfhörerarten nahezu identisch. Der durchschnittliche Winkel ist bei der Verwendung von In-Ear-Kopfhörern sogar geringer. Durch diese Erkenntnis lässt sich sagen, dass es egal ist, mit welcher Art von Kopfhörern die Studie durchgeführt wurde, da diese keinen Einfluss auf das Ergebnis hatten.
Der Auswertung kann außerdem der Einfluss des üblicherweise verwendeten Betriebssystems auf die Leistung der Teilnehmer entnommen werden. Hier ist zu sehen, dass Personen die normalerweise auch ein iPhone nutzen, mit der Anwendung einen besseren Winkel als Personen, die sonst ein Endgerät mit Android verwenden, erzielten. Betrachtet man die erzielte Durchschnittszeit, lässt sich das Gegenteil feststellen. Teilnehmer die üblicherweise ein Androidgerät verwenden, erzielten hier bessere Werte. Insgesamt gesehen fällt es daher schwer ein Urteil über den Einfluss des üblicherweise verwendeten Betriebssystems zu fällen. Die Teilnehmer, welche ansonsten Smartphones mit Android verwenden, durften allerdings keine größeren Probleme, welche die erzielten Ergebnisse negativ beeinflussten, bei der Verwendung und Bedienbarkeit der iOS App haben.

7.6 Aufgetretene Probleme und Feedback

Während der Durchführung der Studie traten ein paar wenige Probleme auf. Zwei Teilnehmer hatten das Problem, dass nach der Rotation des Smartphones in die *Landscape-Left*-Orientierung nichts passierte. Das heißt die Untersuchung der Lokalisationsfähigkeit

wurde nicht automatisch gestartet. Nach gemeinsamer Fehlersuche mit den jeweiligen Teilnehmern wurde festgestellt, dass diese die Ausrichtungssperre ihres iPhones aktiviert hatten. Dadurch wurde die veränderte Orientierung nicht erkannt und die Untersuchung nicht automatisch gestartet. Durch die Deaktivierung der Ausrichtungssperre konnte dieses Problem schließlich gelöst werden.
Ein weiteres, häufiger aufgetretenes Problem war die Tatsache, dass Teilnehmer bei der Durchführung des ersten Durchgangs anstatt einem Tiergeräusch nur einen stark verzerrten Ton hörten. Diese Problematik konnte durch den Neustart der Anwendung und mit einem Neubeginn des Durchgangs umgangen werden. Bei der Suche nach der Ursache für dieses Problem fiel auf, dass vor allem Teilnehmer, die ein etwas älteres iPhone-Modell verwendeten, diese Schwierigkeiten hatten. Vermutlich wurde durch die schlechtere Leistung dieser Modelle die jeweilige Sounddatei nicht schnell genug oder nicht richtig geladen und dadurch auch nicht korrekt abgespielt. Eine Erklärung für den Umstand, dass beim zweiten Starten der Anwendung alles problemlos funktionierte, wurde allerdings nicht gefunden.
Nach der erfolgreichen Durchführung der Studie gab jeder Teilnehmer ein kurzes Feedback ab. Dieses fiel überwiegend positiv aus. Alle fanden die entwickelte App sehr interessant und hatten viel Spaß während der Studie. Durch die Rückmeldungen wurden zusätzlich Hinweise zur Verbesserung der Anwendung und der Studie gegeben. Viele Teilnehmer meinten, dass sie sich nicht sicher waren, ob sie bereits dreimal die Position eines Geräusches erkannt haben oder nicht. Hier wäre eine kleine Meldung, die darauf hinweist, oder eine automatische Beendigung der Untersuchung sehr hilfreich. Außerdem wurde bei den Durchgängen mit zwei Tiergeräuschen bemängelt, dass die Meldung, die vorgibt welche Position gefunden werden muss, zu kurz angezeigt wird. Einige Teilnehmer hätten auch gerne innerhalb der Studie eine andere Szene ausgewählt, um die Durchführung abwechslungsreicher zu gestalten. Diese Anmerkungen sind sehr hilfreich, um eine eventuelle weitere Studie zu optimieren.

8

Zusammenfassung und Ausblick

Dieses abschließende Kapitel gibt zunächst einen Ausblick auf mögliche Verbesserungen und Weiterentwicklungen der entwickelten iOS Anwendung zur Untersuchung der menschlichen Lokalisationsfähigkeit. Im darauffolgenden Abschnitt werden nochmals alle wichtigen Punkte der kompletten Ausarbeitung zusammengefasst.

8.1 Zusammenfassung

Innerhalb dieser Thesis wurde eine Anwendung zur Untersuchung der menschlichen Lokalisationsfähigkeit für das mobile Betriebssystem iOS vorgestellt. Dabei wurden zunächst die Grundlagen des Richtungshörens sowohl aus medizinischer als auch aus technologischer Sicht näher betrachtet. Außerdem wurden diverse Anwendungen, die ebenfalls 3D-Sound beinhalten oder sich mit Richtungshören beschäftigen, vorgestellt. In einem weiteren Kapitel wurden anschließend alle Anforderungen an die zu entwickelnde iOS Anwendung aufgelistet und näher beschrieben. Aus den gesammelten Anforderungen wurde in einem weiteren Schritt eine Konzeption der Anwendung erstellt. Dabei stand die Erstellung von Mockups, einem Komponentendiagramm, einem Datenmodell und einem Klassendiagramm im Vordergrund.
Anschließend wurde im Hauptteil dieser Ausarbeitung auf die Implementierung der App eingegangen. Die wichtigsten verwendeten Technologien sowie die Programmierung von wichtigen Komponenten und dabei aufgetretene Probleme wurden innerhalb dieses Kapitels genauer beschrieben. Nach der Implementierung fand ein Abgleich der entwickelten Anwendung mit den zu Beginn definierten Anforderungen statt.
Schließlich wurde eine Studie druchgeführt, in welcher die Funktionsweise und der Einfluss durch die Anzahl der zu hörenden Geräuschen auf die erzielte Genauigkeit und Geschwindigkeit bei der Untersuchung der Lokalisationsfähigkeit überprüft wurden. Zusätzlich fand ein Vergleich zwischen den Ergebnissen der ausgewerteten Studie mit

Ergebnissen von älteren Studien aus früheren Ausarbeitungen statt. Dabei wurde festgestellt, dass der durchschnittlich erzielte Winkel größer als bei einer älteren iOS App war, aber die durchschnittlich benötigte Zeit mit den älteren Ergebnissen mithalten konnte. In einem abschließenden Teil wurden die erzielten Ergebnisse diskutiert und bewertet.
Zusammengefasst kann gesagt werden, dass innerhalb dieser Arbeit eine iOS App entwickelt wurde, mit welcher die menschliche Lokalisationsfähigkeit überprüft und trainiert werden kann. Die Anwendung wurde durch die Durchführung einer Studie validiert und kann sowohl auf iPhones als auch auf iPads installiert und verwendet werden.

8.2 Ausblick

Die entwickelte iOS Anwendung kann in der aktuellen Version problemlos zur Untersuchung der menschlichen Lokalisationsfähigkeit verwendet werden. Sie bietet unter anderem einige Einstellungsmöglichkeiten, mit welchen unterschiedlichste Schwierigkeitsgrade bei der Untersuchung konfiguriert werden können. Da während der Entwicklung der App und der Anfertigung dieser Ausarbeitung diverse Ideen zur Weiterentwicklung und Verbesserung der Anwendung entstanden sind, werden in diesem Abschnitt einige Vorschläge näher beschrieben.
Die App wurde so entwickelt, dass ohne größeren Aufwand weitere Szenen mit einer beliebigen Anzahl von Tieren hinzugefügt werden können. Dadurch kann den Nutzern mehr Abwechslung geboten werden. Sie haben so eine größere Auswahl an Szenen und können ihre Lokalisationsfähigkeit mit den unterschiedlichsten Tierlauten überprüfen. Die Nutzung der Anwendung bleibt mit Hilfe dieser Erweiterung, die schnell umgesetzt werden kann, länger spannend und der Nutzer beschäftigt sich ausgiebiger damit. Zusätzlich können ganz andere Arten von Szenen, wie zum Beispiel eine Universität, eine Fußgängerzone oder ein Flughafen, mit passenden Geräuschen hinzugefügt werden. Dadurch kann das Nutzererlebnis nochmals gesteigert und auftretende Langeweile verhindert werden.
Eine weitere Möglichkeit zur Weiterentwicklung ist die Gamifizierung der Anwendung. Jede Szene könnte verschiedene Level haben, durch welche sich der Nutzer spielen muss. Mit höherem Level könnte die Schwierigkeit der Untersuchung zunehmen. Zum Beispiel wäre es möglich, die zu treffende Genauigkeit bei der Suche nach der richtigen Position

eines Tiergeräusches zu erhöhen. Das heißt, dass Nutzer die Position eines Geräuschs immer genauer erkennen müssten. Zusätzlich wäre eine Änderung der Lautstärke des Tierlautes oder die Hinzunahme eines beziehungsweise mehrerer Hintergrundgeräusches/e eine weitere Option. Durch unterschiedliche Variationen der in Abschnitt 3.2 vorgestellten Einstellungsmöglichkeiten wäre die Umsetzung von zahlreichen Levels denkbar.
Dem Nutzer könnte beispielsweise zunächst nur eine Szene mit wenigen Geräuschen zur Verfügung gestellt werden. Nachdem die Anwendung entweder eine gewisse Zeit lang verwendet oder eine bestimmte Anzahl an Levels erfolgreich beendet wurde, könnten weitere Szenen und weitere Geräusche freigeschaltet werden. Eine Erhöhung der Konzentration des Anwenders während der Untersuchung wäre so möglich, da dieser bei einem erfolgreichen Abschluss eines Levels eine Belohnung in Aussicht gestellt bekommt und diese auch erreichen will. Außerdem könnte eine Highscore zur App hinzugefügt werden. Mit Hilfe dieser könnten sich die Nutzer mit anderen messen und ihre Leistungen vergleichen. Zusammengefasst würde mit dieser (größeren) Weiterentwicklung das Erlebnis bei der Verwendung der App enorm gesteigert werden.
Wie bereits in Abschnitt 4.2.1 erwähnt wurde, könnte die Speicherung der aufgezeichneten Daten nicht innerhalb einer Datei sondern in einer Datenbank erfolgen. Die von den Nutzern erzielten Resultate könnten, mit einer Einwilligung, automatisch in der Datenbank gespeichert werden. Somit wäre die Durchfühung von weiteren interessanten Auswertungen zur Funktionsweise der App und zur Leistung der unterschiedlichen Nutzer möglich. So könnte auch überprüft werden, ob kleinere Verbesserungen der Applikation, wie zum Beispiel beim Abspielen eines Sounds, etwas gebracht haben und die Nutzer anschließend bessere Ergebnisse erzielen. Da zum Beispiel das Erkennen eines Geräusches direkt vor oder hinter einer Person den Nutzern schwer fällt, könnte die Lautstärke des Geräusches reduziert werden, wenn sich dieses direkt hinter einem befindet. Durch gespeicherte Daten vor und nach dieser Anpassung wäre eine Untersuchung des erzielten Erfolges dieser Änderung durchaus denkbar.
Eine weitere durchführbare Verbesserung der App wäre die Ermöglichung einer korrekten Funktionsweise in allen vorstellbaren Orientierungen eines iPhones. Die aktuelle Anwendung funktioniert nur in der sogenannten *Landscape-Left*-Orientierung. Die Unterstützung der *Landscape-Right*- und der *Portrait*-Orientierung könnte die Benutzung der App vereinfachen.
Die Teilnehmer der durchgeführten Studie machten durch ihr Feedback weitere Vorschläge für die Optimierung der App und einer eventuellen weiteren Studie. Wie bereits

in Abschnitt 7.6 erwähnt wurde, sollte die Meldung, die anzeigt, von welchem Geräusch die Position gefunden werden soll, länger zu sehen sein. Falls dieser Hinweis nicht richtig gelesen oder überhaupt nicht gelesen wurde, hat ein Benutzer der Anwendung noch die Chance nachzuschauen, welche Position er finden muss. Im Falle einer weiteren Studie wäre es sehr wichtig, dem Teilnehmer anzuzeigen wie oft er bereits die Vermutung der Position eines Geräusches abgegeben hat. Dadurch hätten die Teilnehmer eine Kontrollmöglichkeit, dass sie bereits drei mal eine Position gesucht haben und sie den jeweiligen Durchgang beenden können. Dies könnte auch durch eine automatische Beendigung der Untersuchung nach dreimaliger Erkennung einer Position gelöst werden.

Literaturverzeichnis

[1] 148Apps.com. Audio Defence: Zombie Arena. http://media.148apps.com/screenshots/804041240/us-iphone-2-audio-defence-zombie-arena.jpeg, 2014. Besucht am 03.03.2017.

[2] Flavius Alecu. Signed angle between two 2d vectors. http://flaviusalecu.com/post/1071220798, 2016. Besucht am 28.11.2016.

[3] Apple. Swift. http://www.apple.com/de/swift/, 2016. Besucht am 10.01.2017.

[4] Apple. Swift. https://swift.org/about/#swiftorg-and-open-source, 2016. Besucht am 06.01.2017.

[5] Apple. Swift 3. https://developer.apple.com/swift/, 2016. Besucht am 10.01.2017.

[6] Apple. Swift and Objective-C in the Same Project. https://developer.apple.com/library/content/documentation/Swift/Conceptual/BuildingCocoaApps/MixandMatch.html, 2016. Besucht am 10.01.2017.

[7] Apple. Swift Playgrounds. http://www.apple.com/swift/playgrounds/, 2016. Besucht am 06.01.2017.

[8] Apple. Xcode. https://developer.apple.com/xcode/ide/, 2016. Besucht am 06.01.2017.

[9] Apple. Xcode 8. https://developer.apple.com/xcode/ide/images/ide-large_2x.png, 2016. Besucht am 21.12.2016.

[10] Apple. Xcode 8. https://developer.apple.com/xcode/interface-builder/images/interface-builder.png, 2016. Besucht am 21.12.2016.

[11] Apple. TestFlight Beta Testing. https://developer.apple.com/testflight/, 2017. Besucht am 20.02.2017.

[12] Durand R Begault, Elizabeth M Wenzel, and Mark R Anderson. Direct comparison of the impact of head tracking, reverberation, and individualized head-related transfer functions on the spatial perception of a virtual speech source. *Journal of the Audio Engineering Society*, 49(10):904–916, 2001.

[13] Jens Blauert. Sound localization in the median plane. *Acta Acustica united with Acustica*, 22(4):205–213, 1969.

[14] Jörg Breithut. In dieser App kommen Zombies durch den Kopfhörer. `http://www.spiegel.de/netzwelt/games/audio-defence-zombie-arena-fuer-ios-a-1000419.html`, 2014. Besucht am 15.02.2017.

[15] CocoaPods. What is CocoaPods. `https://cocoapods.org`, 2016. Besucht am 10.01.2017.

[16] DevelopmentSquared. 3D-Sound-Illusionen. `https://itunes.apple.com/de/app/3d-sound-illusionen/id421894061?mt=8`, 2010. Besucht am 15.02.2017.

[17] Andy Field. *Discovering statistics using SPSS*. Sage publications, 2009.

[18] Michael A Gerzon. Ambisonics in multichannel broadcasting and video. *Journal of the Audio Engineering Society*, 33(11):859–871, 1985.

[19] Google. Downloads and Samples. `https://developers.google.com/vr/ios/download`, 2016. Besucht am 10.01.2017.

[20] Google. Google Cardboard. `https://vr.google.com/cardboard/`, 2016. Besucht am 10.01.2017.

[21] Google. Google Daydream. `https://vr.google.com/daydream/`, 2016. Besucht am 10.01.2017.

[22] Google. Google VR SDK for iOS. `https://developers.google.com/vr/ios/`, 2016. Besucht am 10.01.2017.

[23] Google. Spatial Audio. `https://developers.google.com/vr/concepts/spatial-audio`, 2016. Besucht am 10.01.2017.

[24] Jack Hebrank and D Wright. Are two ears necessary for localization of sound sources on the median plane? *The Journal of the Acoustical Society of America*, 56(3):935–938, 1974.

[25] Jack Hebrank and D Wright. Spectral cues used in the localization of sound sources on the median plane. *The Journal of the Acoustical Society of America*, 56(6):1829–1834, 1974.

[26] Fabian Henkel. *Implementation and evaluation of a mobile web application for auditory stimulation of chronic Tinnitus patients*. PhD thesis, Institute of Databases

and Information Systems, 2015.

[27] Justinmind. Justinmind. `https://www.justinmind.com`, 2016. Besucht am 23.12.2016.

[28] Fakheredine Keyrouz and Klaus Diepold. An enhanced binaural 3D sound localization algorithm. In *Signal Processing and Information Technology, 2006 IEEE International Symposium on*, pages 662–665. IEEE, 2006.

[29] Jemima Kiss. Papa Sangre: The sonic iPhone horror game you've been looking for. `https://www.theguardian.com/technology/pda/2010/dec/20/papa-sangre-game-audio`, 2010. Besucht am 03.03.2017.

[30] M. Kofler. *Swift 2: das umfassende Praxisbuch : [Syntax, Grundlagen, App-Entwicklung] : [Objekt- und protokollorientierte Anwendungsentwicklung] : [mit vielen Beispiel-Apps für iOS und OS X]*. Rheinwerk Computing. Rheinwerk, 2015.

[31] National Acoustic Laboratories. About Sound Storm. `https://capd.nal.gov.au/sound-storm-about.shtml`, 2016. Besucht am 15.02.2017.

[32] National Acoustic Laboratories. About Sound Storm. `https://capd.nal.gov.au/images/Sound%20Storm%20Select%20World%20Sreen.png`, 2016. Besucht am 27.02.2017.

[33] National Acoustic Laboratories. About Sound Storm. `https://capd.nal.gov.au/images/Sound%20Storm%20Correct%20Response%20Screen.PNG`, 2016. Besucht am 27.02.2017.

[34] Maruricio Lumbreras and Jaime Sánchez. Interactive 3D sound hyperstories for blind children. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 318–325. ACM, 1999.

[35] Ewan A Macpherson and John C Middlebrooks. Listener weighting of cues for lateral angle: the duplex theory of sound localization revisited. *The Journal of the Acoustical Society of America*, 111(5):2219–2236, 2002.

[36] Allen William Mills. On the minimum audible angle. *The Journal of the Acoustical Society of America*, 30(4):237–246, 1958.

[37] Markus Noisternig, Thomas Musil, Alois Sontacchi, and Robert Höldrich. A 3D real time Rendering Engine for binaural Sound Reproduction. Georgia Institute of Technology, 2003.

[38] Simon R Oldfield and Simon PA Parker. Acuity of sound localisation: a topography of auditory space. II. Pinna cues absent. *Perception*, 13(5):601–617, 1984.

[39] Tilen Potisk. Head-Related Transfer Function. *University of Ljubljana, Faculty of Mathematics and Physics*, 2015.

[40] Quicktech. Audio Defence: Zombie Arena. `http://static.trustedreviews.com/94/00002f259/ac27/AudioDefense1.jpg`, 2014. Besucht am 27.02.2017.

[41] Lord Rayleigh. XII. On our perception of sound direction. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 13(74):214, 1907.

[42] Suzanne K Roffler and Robert A Butler. Localization of tonal stimuli in the vertical plane. *The Journal of the Acoustical Society of America*, 43(6):1260–1266, 1968.

[43] Jaime Sánchez and Mauricio Sáenz. 3D sound interactive environments for blind children problem solving skills. *Behaviour & Information Technology*, 25(4):367–378, 2006.

[44] Papa Sangre. Audio Defence (Official launch trailer). `https://www.youtube.com/watch?v=xn6G2fxlsCU`, 2014. Besucht am 03.03.2017.

[45] Johanna Barbara Sattler, André Klußmann, Birgit Arnold-Schulz-Gahmen, Almuth Vasterling, Hubert Wagner, and Bernd Hartmann. Händigkeit–Bedeutung und Untersuchung. November 2014.

[46] Marc Schickler, Rüdiger Pryss, Manfred Reichert, Martin Heinzelmann, Johannes Schobel, Berthold Langguth, Thomas Probst, and Winfried Schlee. Using Wearables in the Context of Chronic Disorders - Results of a Pre-Study. In *29th IEEE Int'l Symposium on Computer-Based Medical Systems*, pages 68–69, June 2016.

[47] Marc Schickler, Rüdiger Pryss, Manfred Reichert, Johannes Schobel, Berthold Langguth, and Winfried Schlee. Using Mobile Serious Games in the Context of Chronic Disorders - A Mobile Game Concept for the Treatment of Tinnitus. In *29th IEEE Int'l Symposium on Computer-Based Medical Systems (CBMS 2016)*, pages 343–348, June 2016.

[48] Marc Schickler, Johannes Schobel, Rüdiger Pryss, and Manfred Reichert. Mobile Crowd Sensing? A New way of collecting data from trauma samples? In *XIV Conference of European Society for Traumatic Stress Studies (ESTSS) Conference*, page 244, June 2015.

[49] Johannes Schobel, Rüdiger Pryss, Marc Schickler, and Manfred Reichert. Towards Flexible Mobile Data Collection in Healthcare. In *29th IEEE International Symposium on Computer-Based Medical Systems (CBMS 2016)*, pages 181–182, June 2016.

[50] Johannes Schobel, Marc Schickler, Rüdiger Pryss, and Manfred Reichert. Process-Driven Data Collection with Smart Mobile Devices. In *10th International Conference on Web Information Systems and Technologies (Revised Selected Papers)*, number 226 in LNBIP, pages 347–362. Springer, 2015.

[51] Stanley Smith Stevens and Edwin B Newman. The localization of actual sources of sound. *The American Journal of Psychology*, 48(2):297–306, 1936.

[52] Otis C Trimble. The theory of sound localization: a restatement. *Psychological Review*, 35(6):515, 1928.

[53] Martin Weidhaas. *Implementation and evaluation of a mobile Windows-application for auditory stimulation of chronic tinnitus patients.* PhD thesis, Institute of Databases and Information Systems, 2015.

[54] Bernd Wener. App der Woche: Papa Sangre – mit den Ohren spielen. `http://www.gamersglobal.de/news/31486/app-der-woche-papa-sangre-mit-den-ohren-spielen`, 2011. Besucht am 03.03.2017.

[55] Bernd Wener. App der Woche: Papa Sangre – mit den Ohren spielen. `http://www.gamersglobal.de/sites/gamersglobal.de/files/userupload/user1886/AppDerWoche/adw_PapaSangre.jpg`, 2011. Besucht am 03.03.2017.

[56] Frederic L Wightman and Doris J Kistler. Headphone simulation of free-field listening. I: stimulus synthesis. *The Journal of the Acoustical Society of America*, 85(2):858–867, 1989.

[57] Frederic L Wightman and Doris J Kistler. Headphone simulation of free-field listening. II: Psychophysical validation. *The Journal of the Acoustical Society of America*, 85(2):868–878, 1989.

[58] William A Yost and George Gourevitch. *Directional hearing.* Springer, 1987.

A

Klassendiagramm

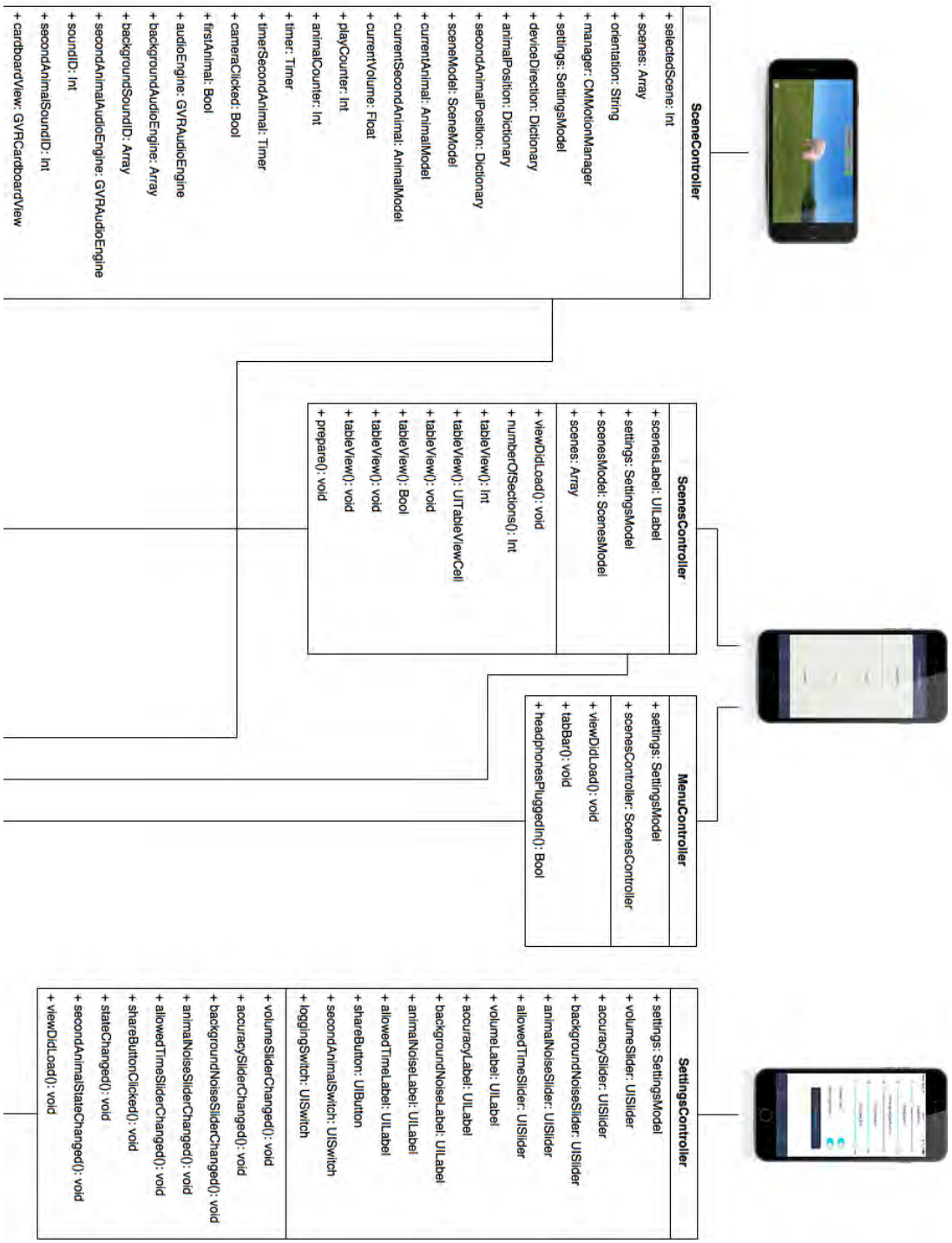


Abbildung A.1: Klassendiagramm Teil 1

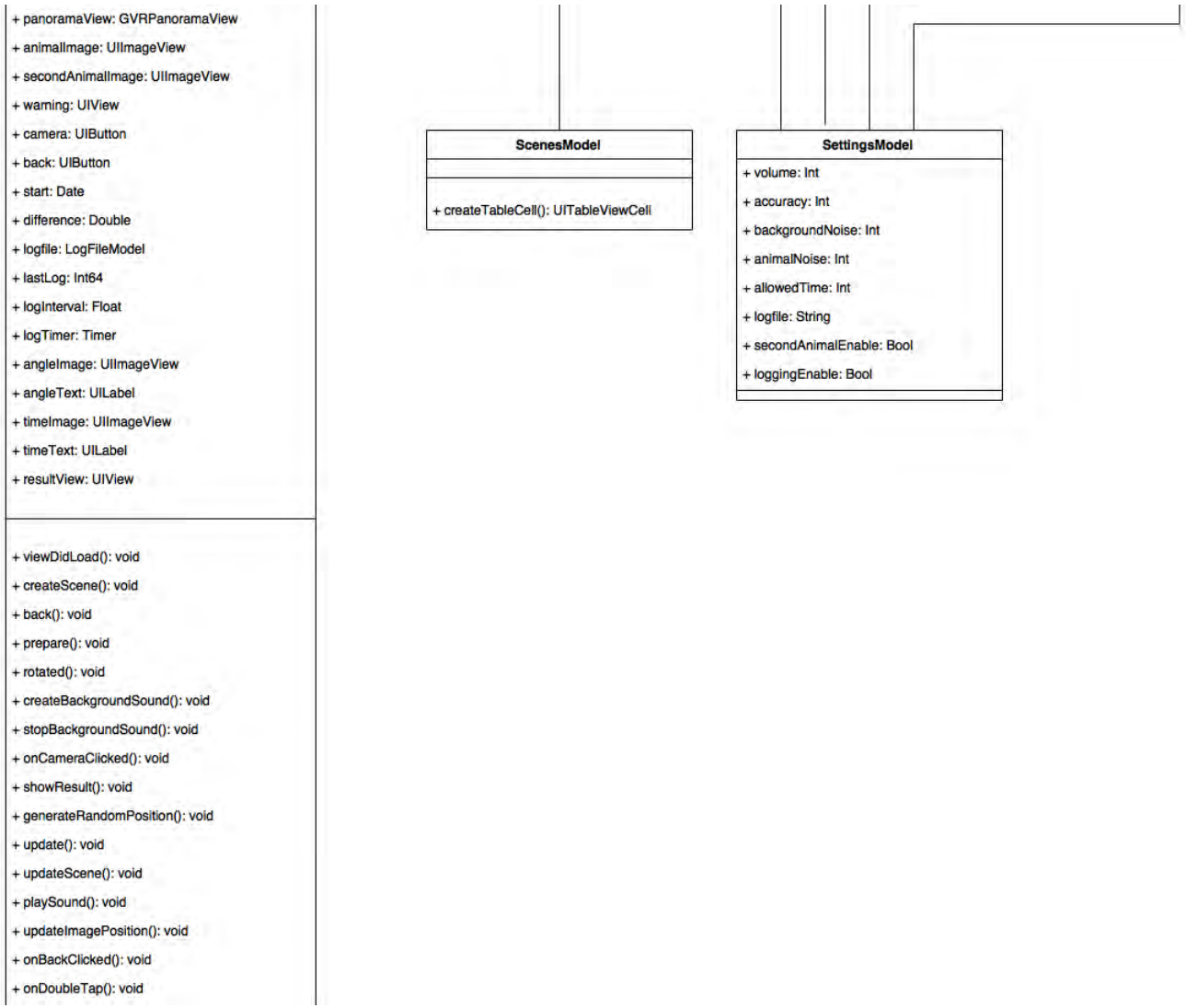


Abbildung A.2: Klassendiagramm Teil 2

Abbildung A.3: Klassendiagramm Teil 3

B

Studienanleitung

Studie zur Untersuchung der menschlichen Lokalisationsfähigkeit mit Hilfe einer iOS-App

Erklärung

Innerhalb meiner Masterthesis habe ich eine iOS-App zur Untersuchung der menschlichen Lokalisationsfähigkeit entwickelt. Diese App soll nun durch eine kleine Studie validiert werden. Insgesamt ist zunächst ein Fragebogen zu beantworten, anschließend müssen drei Durchgänge, die später näher beschrieben werden, durchgeführt werden.
Um die Funktionsweise der App besser zu verstehen, kann man sich folgendes Szenario vorstellen: Man läuft spazieren und sieht ein Tier das man unbedingt fotografieren will. Sofort wird das Smartphone herausgeholt und die Kamera gestartet. Doch diese ist leider defekt und man kann nicht mehr sehen, was man fotografiert. Um nun ein Foto des Tieres machen zu können, muss man mit Hilfe seiner Ohren erkennen, wo sich das Tier befindet. Anschließend hält man das Smartphone in die vermutete Richtung und drückt den Auslöser.
Genau dieses Szenario soll mit Hilfe der App simuliert und dadurch deine Lokalisationsfähigkeit getestet werden.

Du benötigst:
- Dein iPhone oder iPad
- Kopfhörer, am besten On-Ear Kopfhörer
- Einladung von Apple TestFlight (die hast du bereits bekommen)
- Ca. 25 Minuten Zeit

Vorbereitung:
- Apple TestFlight App installieren (https://itunes.apple.com/de/app/testflight/id899247664?mt=8)
- Einladung von Apple TestFlight öffnen und auf „Start Testing" klicken und meine App installieren
- Kopfhörer einstecken
- Lautstärke so laut wie möglich einstellen
- So hinstellen oder –sitzen damit du dich frei bewegen kannst

Fragebogen

Bitte beantworte zunächst folgende Fragen und schicke mir die Antworten per E-Mail an **marcel.erath@uni-ulm.de**. Gebe im Betreff bitte deinen Vor- und Nachnamen so wie das Wort „Fragebogen" an.

Beispiel Betreff: **Max Mustermann Fragebogen**

Fragen:

1. Wie alt bist du?
2. Bist du männlich oder weiblich?
3. Hast du Tinnitus oder eine andere Erkrankung bezüglich deines Gehörs?
4. Wie viel Erfahrung hast du mit mobilen Geräten?

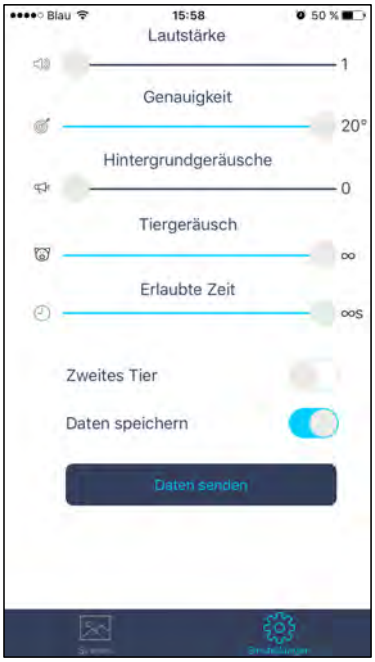
wenig				viel
☐	☐	☐	☐	☐
1	2	3	4	5

5. Welches Smartphone verwendest du üblicherweise?
6. Verwendest du On- oder In-Ear Kopfhörer?
7. Wie viel Erfahrung hast du mit Videospielen?

wenig				viel
☐	☐	☐	☐	☐
1	2	3	4	5

1. Durchgang: 3x ein Tierlaut ohne Hintergrundgeräusch

- Zunächst die Anleitung zu diesem Durchgang komplett durchlesen
- App öffnen
- Einstellungen auswählen
- Folgende Werte (s. Abbildung) einstellen

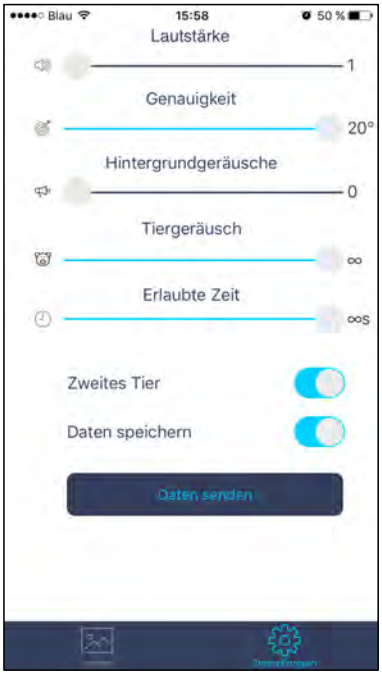


- Auf Szenen klicken
- Szene „Bauernhof" auswählen
- iPhone zum Starten nach links drehen
- Tierlaut ertönt, dessen Position muss erkannt werden
- iPhone in die vermutete Richtung halten
- mit Klick auf Kamerabutton Position bestätigen
- durch Doppelklick auf das Display nächster Tierlaut abspielen
- Vorgang 3x wiederholen
- Auf „Zurück" klicken

- E-Mail senden:
  - Einstellungen wählen
  - Auf den Button „Daten senden" klicken
  - Im erscheinenden Fenster „Mail" auswählen
  - Betreff im Format **Vor- und Nachname + durchgeführter Durchgang**
    Beispiel: Max Mustermann 1. Durchgang
  - E-Mail an **marcel.erath@uni-ulm.de** senden
  - Zum nächsten Durchgang gehen

2. Durchgang: 3x zwei Tierlaute ohne Hintergrundgeräusch

- Zunächst die Anleitung zu diesem Durchgang komplett durchlesen
- Einstellungen auswählen
- Folgende Werte (s. Abbildung) einstellen

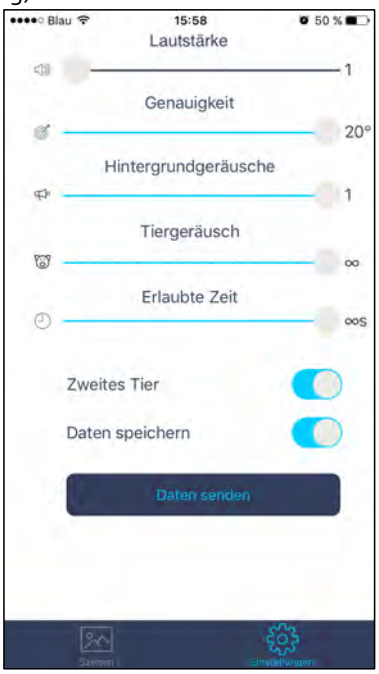


- Auf Szenen klicken
- Szene „Bauernhof" auswählen
- iPhone zum Starten nach links drehen
- es ertönen zwei Tierlaute, die beide erkannt werden müssen. Welche Position zuerst gefunden werden muss, wird angezeigt.
- iPhone in die vermutete Richtung halten
- mit Klick auf Kamerabutton Position bestätigen
- zweiter Tierlaut finden
- durch Doppelklick auf das Display nächster Tierlaut abspielen
- Vorgang 3x wiederholen
- Auf „Zurück" klicken

- E-Mail senden:
  - Einstellungen wählen
  - Auf den Button „Daten senden" klicken
  - Im erscheinenden Fenster „Mail" auswählen
  - Betreff im Format **Vor- und Nachname + durchgeführter Durchgang**
    Beispiel: Max Mustermann 2. Durchgang
  - E-Mail an **marcel.erath@uni-ulm.de** senden
  - Zum nächsten Durchgang gehen

3. Durchgang: 3x zwei Tierlaute mit Hintergrundgeräusch

- Zunächst die Anleitung zu diesem Durchgang komplett durchlesen
- Einstellungen auswählen
- Folgende Werte (s. Abbildung) einstellen



- Auf Szenen klicken
- Szene „Bauernhof" auswählen
- iPhone zum Starten nach links drehen
- Hintergrundgeräusch ertönt
- es ertönen zwei Tierlaute, die beide erkannt werden müssen. Welche Position zuerst gefunden werden muss, wird angezeigt.
- iPhone in die vermutete Richtung halten
- mit Klick auf Kamerabutton Position bestätigen
- zweiter Tierlaut finden
- durch Doppelklick auf das Display nächster Tierlaut abspielen
- Vorgang 3x wiederholen
- Auf „Zurück" klicken

- E-Mail senden:
  - Einstellungen wählen
  - Auf den Button „Daten senden" klicken
  - Im erscheinenden Fenster „Mail" auswählen
  - Betreff im Format **Vor- und Nachname + durchgeführter Durchgang**
    Beispiel: Max Mustermann 3. Durchgang
  - E-Mail an **marcel.erath@uni-ulm.de** senden

Vielen Dank für deine Teilnahme an meiner Studie!!

Marcel

5